RADIANCEKIT

User Manual

Photorealistic 3D Reconstruction
via Gaussian Splatting

Version 1.5.0 · macOS 26.0+ · May 2026

BJOERN KINDLER · KINDLER-DEV.DE

CONTENTS

Overview

READING GUIDE

How to read this manual

Every entry in this manual follows the same scheme. On the left-hand side you'll find the operating paths and technical details; on the right, in a warm sidebar, you'll always find the plain-language explanation. Small icons at the start of each line tell you at a glance what kind of information is coming next.

THE FOUR ICONS

Wo (Where). The concrete click path through the app — menu bar, Inspector section, or Simple Mode step. Associated keyboard shortcuts also live here. The icon is a map pin and shows where the feature sits in the user interface.

Details. Default values, value ranges, and code paths. You'll meet this mainly in the training settings, which are not menu items but numeric parameters. The icon shows a small specification card.

Technisch (Technical). What the feature does internally, which parameters take effect, what it reacts to, and what side effects it has. For readers who want to understand what is happening behind the scenes. The icon is a slider block and stands symbolically for the knobs under the hood.

Einfach gesagt (Plain English). The core message in clear words — no jargon, no code. Read this section first if you just want to know quickly what a feature is for and when you'd use it. The icon is a speech bubble and stands for "in a nutshell". This column is always set on a warm sand tone so the eye finds it immediately.

CHAPTER COLORS

Each chapter has its own accent color which you'll recognize from the ID badge (for example M1) to the left of every entry title and from the small icons in front of them. As you flip through, you can see at a glance which chapter you're currently in.

1 Menus · 2 Inspector · 3 Settings · 4 Aux Windows · 6 Training

7 Presets · 8 Exports · 9 SfM · 10 Simple Mode

NAVIGATION TIPS

Quick start. If you're only interested in operating the app, jump straight to **Chapter 10 — Simple Mode**. That's the guided variant with four steps and requires no prior knowledge.

Going deeper. Chapter 2 — Inspector and **Chapter 7 — Presets** explain the controls and the preconfigured quality profiles available to you in Expert Mode.

Looking things up. The table of contents and the PDF full-text search help you find a specific feature. You don't have to read the manual cover to cover.

CHAPTER

Introduction — What you should know

What is RadianceKit?

RadianceKit is a native macOS app that turns a series of ordinary photos or a video into a walkable 3D reconstruction. The input is, for example, 50 to 500 shots that you took around an object, through a room, or across a landscape. The output is what's known as a Gaussian-Splatting scene — a 3D model you can view on the Mac in real time from any angle, that can be exported and embedded on websites, and that, in its main respects, looks photoreal.

The app runs entirely locally on your Mac — no images are uploaded to the cloud, no login is required, and there's no subscription. It makes intensive use of the GPU on your Apple-silicon Mac (M series): a full training run can take anywhere from two minutes to several hours depending on the scene and preset. While the computation is running, you can keep working on your Mac as usual; RadianceKit continues in the background and notifies you when the result is ready.

There are two operating modes: *Simple Mode* (Einsteiger-Modus) guides you through the workflow Import → Choose Preset → Training → Export in four steps. *Expert Mode* (Experten-Modus) opens a large Inspector with every knob, a live preview window, and diagnostic charts. You can switch between modes at any time; the data in the scene is preserved.

What is Gaussian Splatting?

Gaussian Splatting (often abbreviated as *3DGS* or simply *Splatting*) is a relatively new method for photorealistic 3D representation, introduced in 2023 in a paper from Graz and INRIA. The idea: instead of modelling a scene as a classical polygon mesh (triangles) or a voxel grid, it is assembled from millions of small, soft 3D blobs — each individual blob is a 3D Gaussian distribution (hence the name) with its own position, size, shape, color, and transparency. These blobs are trained so that, taken together from every viewing angle of your input photos, they yield the correct image.

In practice, this means Gaussian Splatting can represent reflections, highlights, soft foliage, hair, or curtains the way classical 3D modelling cannot — or can only at enormous

cost. In return, the result is not an editable 3D model in the classical sense — you can't simply move a single wall or relocate a vase. It is more of a *frozen capture* of the space that you can move through freely. For many applications — architectural visualization, product presentation, virtual tours, forensics, cultural heritage — that is exactly the right strength.

Two steps are needed to turn the input images into a 3D scene. First, the app computes via a process called *Structure-from-Motion (SfM)* where your camera was standing for each photo. As a byproduct, this produces a rough point cloud of the scene. Then the actual Gaussian-Splatting training begins: starting from this rough cloud, the millions of 3D blobs are gradually distributed, grown, refined, and re-tuned in position and color until they yield the matching image from all input viewing angles.

You don't have to know any of this to use RadianceKit. Simple Mode hides these steps completely. But if you want to understand what the diagnostic numbers in Expert Mode (iteration, loss, Gaussians, SSIM …) mean, or why some scenes turn out nicer than others, the later chapters of the manual will give you the answers.

CHAPTER

Chapter 1 — Menu Bar

The RadianceKit menu bar organizes every function that doesn't sit directly in the main window or the Inspector. These are mainly actions that affect the whole scene (Open, Save, New Project), control training (Start, Pause, Continue), operate the viewport (auto-rotation, screenshot, background color) and trigger exports to various 3D and media formats. On top of that there are jump points to all helper windows (User Guide, Pareto Dashboard, Holdout Analysis, BayesOpt Console).

Keyboard shortcuts are listed to the right of each menu entry. Conventions: ⌘ is the Command key (Apple key), ⇧ is Shift, ⌥ is Option (Alt), and ⌃ is Control. Example: ⇧⌘T means Shift+Command+T. All shortcuts documented here are additionally listed in a dedicated overview window under `Help → Keyboard Shortcuts` (⌘/).

The following 42 entries are documented in inventory order (M1–M42), grouped by the top-level menu they belong to. All entries have been verified against the current code state in (lines 175–477). No entries have been removed or superseded relative to the inventory; a new Edit menu entry (Cmd-Z for "Remove Image") is provided by the system NSUndoManager framework and therefore does not appear in the RadianceKitApp code (see note at the end of the chapter).

File Menu

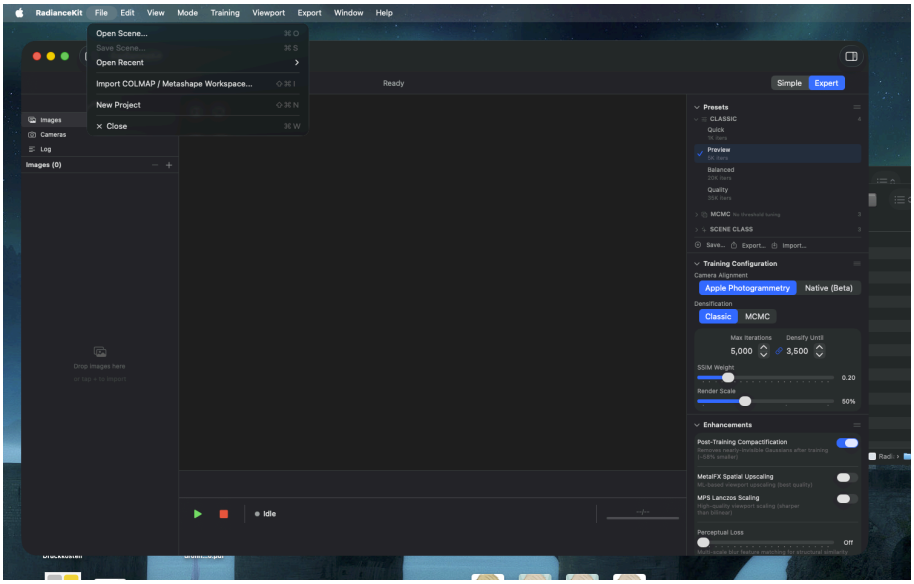

Figure 1: File menu expanded — entries M1 through M6

The File menu replaces Apple's standard "New Window" entry with project-specific actions. It covers loading/saving scenes, a dynamic Recent list, workspace import, and a hard reset to an empty state.

M1 File > Open Scene…

WHERE

Menu bar → File → Open Scene… (⌘O).

TECHNICAL

Opens a file dialog for the formats `RadianceScene` bundle, `.ply`, `.splat` and `.spz`. Single selection, can show both files and directories (for the bundle format). After a successful selection the path is added to the Recent list and the scene is loaded asynchronously — the previous one is replaced and the training pipeline is initialized with the loaded state. PLY/SPZ/Splat files are read via their respective format loaders; the `.radiancescene` bundle is a directory with manifest, cloud snapshot, and SfM results.

IN PLAIN WORDS

This is how you reload an already-trained scene back into the app. Works with RadianceKit's own format and with the standard formats PLY, SPLAT, and SPZ that other splatting programs produce. Use this when, e.g., you've trained a scene overnight and want to continue or export the next day. On opening, the current state in the main window is replaced — so save first if the current scene still matters to you. The path automatically lands in "Open Recent" (M3) so you can get back to it faster next time.

M2 File > Save Scene…

WHERE

Menu bar → File → Save Scene… (⌘S).

TECHNICAL

Opens a file save dialog with the content type `RadianceScene` bundle and a prefilled filename `scene.radiancescene`. Writes a directory package with `manifest.json`, the serialized Gaussian cloud (PLY snapshot) and a dump of the SfM result, so that continue training also works after reopening. The entry is disabled as long as no Gaussians exist yet. Does not save to the training-logs path but to wherever the save dialog points — typically under `~/Documents/`.

IN PLAIN WORDS

Saves your current scene as a file (more precisely: as a package folder that looks like a file). Only after this can you later reopen the scene with "Open Scene…" (M1). The package contains both the Gaussian cloud and the SfM result, so you can also append continue training (M12–M14) later. As long as you haven't finished a training run, the entry is grayed out. The default name is `scene.radiancescene` — but you can give it your own name in the Save dialog.

M3 File > Open Recent > [scene names]

WHERE

Menu bar → File → Open Recent → (list).

TECHNICAL

Dynamic submenu generated from a list of recently opened paths (stored in the settings). Each list entry is labeled with the filename and loaded on click. If the list is empty, the disabled label "No Recent Scenes" appears instead. Apple-style, the list holds the N most recently opened scenes — the limit is enforced when writing to the settings, not in the menu builder itself.

IN PLAIN WORDS

Here you see the most recently opened scenes and can jump back in with a click without going through the file dialog. If you've just started, the list is empty and grayed out in the menu. Every scene you open via "Open Scene..." (M1) automatically lands in this list. If the list ever gets too full or you want to clear it for privacy reasons, use "Clear Recent" (M4).

M4 File > Open Recent > Clear Recent

WHERE

Menu bar → File → Open Recent → Clear Recent.

TECHNICAL

Clears the Recent list in the settings. Takes effect immediately, without a confirmation dialog. The entry only appears in the submenu when there are actually entries in the Recent list (it sits below a divider after the paths).

IN PLAIN WORDS

Clears the list of recently opened scenes. Handy when you've been playing around with a test dataset and don't want to see the paths anymore. The scene files themselves are not deleted — only the link in the menu. The action takes effect immediately, without confirmation; afterwards the submenu shows "No Recent Scenes". The entry only shows up if there are actually scenes in the list — when the list is empty, it's invisible.

M5 File > Import COLMAP / Metashape Workspace...

WHERE

Menu bar → File → Import COLMAP / Metashape Workspace... (⇧⌘I).

TECHNICAL

Opens a folder picker. Expects a folder with the COLMAP workspace layout (e.g., `sparse/0/cameras.{bin,txt}` plus `images/`). After selection, a workspace pre-check is run — it detects the three layouts (`sparse/0/`, `sparse/`, root) and whether the reconstruction is binary (`cameras.bin`) or in ETH3D text form (`cameras.txt`). On success the workspace is imported; otherwise only a warning appears in the app log. See also Chapter 9 “SfM Backends”, Q6 for the full pipeline logic.

IN PLAIN WORDS

If you use Metashape, COLMAP, RealityCapture, or a similar piece of software for camera reconstruction and have an export, load the folder here. RadianceKit then skips the SfM stage and starts training directly — which saves hours on large scenes. Drag-and-drop onto the main window works the same way. A folder with the COLMAP layout is expected (i.e., `sparse/0/` with `cameras.*` plus an `images/` folder). More on the supported layouts and workflows is in Chapter 9 “SfM Backends”.

M6 File > New Project

WHERE

Menu bar → File → New Project (⇧⌘N).

TECHNICAL

Checks whether unsaved work is present. If so, a confirmation dialog appears before anything gets lost. If there’s nothing to save, the reset runs immediately — it clears imported images, the SfM result, the Gaussian cloud, training state and all dependent UI indicators. Note: A user-created preset library is preserved because it lives in the app settings and not in the project state.

IN PLAIN WORDS

Resets everything to an empty start — as if you had just freshly opened the app. If you still have unsaved work, the app asks first. Use this when you want to start fresh with a completely different scene. Imported images, SfM result, Gaussian cloud and training state are completely cleared. Your own presets are preserved, though, because they live in the app settings and don’t belong to the scene.

Mode Menu

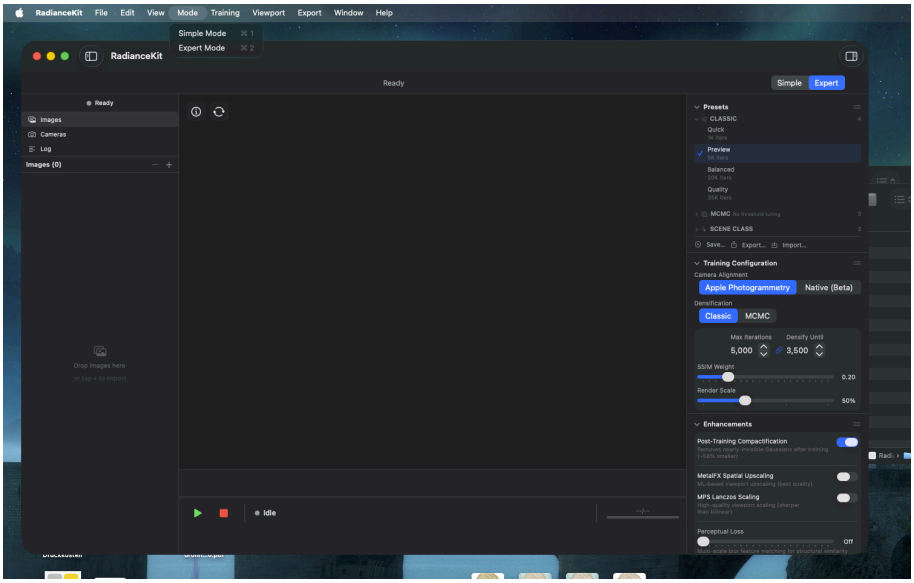

Figure 2: Mode menu with Simple and Expert Mode switches

Two simple switches between the guided Simple Mode (wizard-like, 4 steps) and the full Expert Mode (classic Inspector layout with all controls).

M7 Mode > Simple Mode

WHERE

Menu bar → Mode → Simple Mode (⌘1).

TECHNICAL

Switches the app state to Simple Mode. The main area of the app then shows the guided workflow instead of the Expert layout. The mode state is stored in the settings (see S1 “Default Mode” in Chapter 3 Settings).

IN PLAIN WORDS

Switches to the step-by-step variant where the app guides you through importing, processing, preview, and export. Recommended if you’re just starting out or need a quick result. Most detail controls are hidden — you work with sensible defaults. If you want to dig deeper later, just switch to Expert Mode (M8). Which mode is active when the app starts can be set in the settings (Chapter 3, S1).

M8 Mode > Expert Mode

WHERE

Menu bar → Mode → Expert Mode (⌘2).

TECHNICAL

Switches the app state to Expert Mode. This brings up the full Inspector layout with all sections (Presets, TrainingConfig, Enhancements, Metrics, LossChart, ProjectNavigator). In Expert Mode all training parameters, COLMAP picker, mid-compact toggles, and diagnostics are accessible. The live preview also only works in this mode.

IN PLAIN WORDS

Switches to the full view with all controls. Here you see loss charts in real time, can fine-tune all parameters and manage multiple comparison configs in parallel via presets. Recommended if you want to understand what training is doing internally, or if you want to experiment in a targeted way. The live preview, COLMAP picker, and diagnostics are also only accessible here. If you feel overwhelmed, go back to Simple Mode via M7 — your scene is preserved in the process.

Training Menu

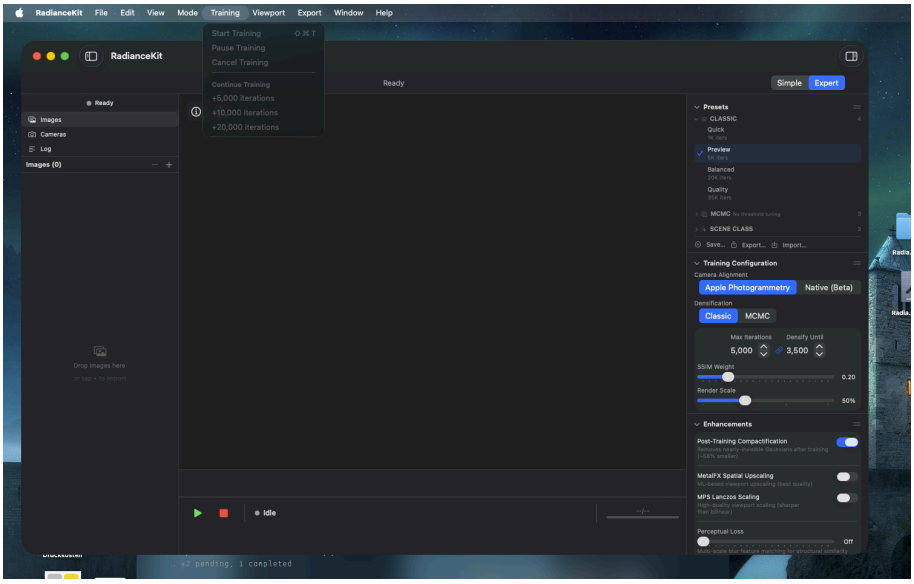

Figure 3: Training menu with Continue submenu — entries M9 through M14

Four actions around the training run: start, pause, cancel, and extend by a fixed iteration count. All three Continue entries are IAP-gated (not clickable in the free trial version).

M9 Training > Start Training

WHERE

Menu bar → Training → Start Training (⇧⌘T).

TECHNICAL

Starts the training pipeline asynchronously. Prerequisite: an SfM result exists and no other pipeline is currently running. Both conditions block the entry if not met. On start, the current configuration values are read, a new JSONL log under `~/Documents/RadianceKit/Logs/training_YYYY-MM-DD_HHmmss.jsonl` is created, and depending on the strategy choice the classic or the MCMC path is taken. The training state switches from "idle" to "training".

IN PLAIN WORDS

Hits the big green button — once you've imported photos and the camera reconstruction is done, the actual Gaussian-Splatting training starts. Let the app run; depending on the preset between 1 minute (Quick) and several hours (MCMC Quality). The entry stays gray as long as no SfM result is available yet or another pipeline is currently running. Each run also writes a log to `~/Documents/RadianceKit/Logs/`, which you can analyze later via the Pareto Dashboard (M40).

M10 Training > Pause Training

WHERE

Menu bar → Training → Pause Training.

TECHNICAL

Pauses the running training. Only enabled when the training state is "training". Pausing stops the iteration loop at the next safe sync point, keeps the full GPU state (Gaussian buffers, optimizer moments, scheduler position) and switches to "paused". Resume happens by pressing it again (the entry title is static — but the app toggles between Pause/Resume in the actual logic). Paused trainings do not survive an app quit; in that case save the scene instead and extend it later via a Continue Training entry (M12–M14).

IN PLAIN WORDS

Briefly halts training without losing progress. Handy when you briefly need the computer for something more important. Click again to resume. Doesn't work across app restarts — if you really want to continue later, end the training with Cancel (M11), save the scene with Save Scene (M2), and then use Continue Training (M12–M14). During the pause the GPU is completely idle; the memory remains allocated, though.

M11 Training > Cancel Training

WHERE

Menu bar → Training → Cancel Training.

TECHNICAL

Cancels the running training. Active when the training state is not “idle”. Sets the cancel flag in the training engine, which cleanly ends the iteration loop at the next sync point, writes the final summary entry into the JSONL log, and resets the state to “idle”. The cloud trained so far is preserved (can be saved or exported) but marked as “cancelled”.

IN PLAIN WORDS

Cancels the running training for good. The current state is preserved — so if after a few thousand iterations you already have a presentable result, you can still export it. If you only want to briefly interrupt, use Pause (M10) instead. In the training log, the run is marked as “cancelled”, and the final loss value is still written out. A cancelled scene can also be continued via Continue Training (M12–M14) later, as long as the app hasn’t been quit in the meantime.

M12 Training > Continue Training > +5,000 iterations

WHERE

Menu bar → Training → Continue Training → +5,000 iterations.

TECHNICAL

Continues training by 5,000 iterations. Active when a completed training run is resumable and the full version is unlocked. Resumability holds when a completed training run exists and the full optimizer state is still in memory. On continue, the Adam moments and the LR scheduler are carried forward, so the continuation behaves like a continuous 25K/45K/60K run rather than a restart. The JSONL log gets a new config entry with the incremental setup. Only available in the full version.

IN PLAIN WORDS

Adds another 5,000 training steps. Use this when the result after the first run is close, but not quite sharp yet. Only works in the paid full version. Unlike a fully new run, the optimizer state is preserved, so the continuation feels like a continuous run. If you need more than 5,000 steps, go straight to M13 (+10,000) or M14 (+20,000).

M13 Training > Continue Training > +10,000 iterations

WHERE

Menu bar → Training → Continue Training → +10,000 iterations.

TECHNICAL

Identical to M12, but with 10,000 additional iterations. Same prerequisites, same LR scheduler path. Recommended when the initial training was run with a mid-tier preset and you want to see a significant quality improvement without restarting the run from scratch.

IN PLAIN WORDS

Extends training by 10,000 steps — the middle of the three available Continue values. Good choice when the first run was okay but you still want to get clearly better. Like M12 and M14, the learning-rate schedule continues seamlessly instead of restarting. Only available in the full version.

M14 Training > Continue Training > +20,000 iterations

WHERE

Menu bar → Training → Continue Training → +20,000 iterations.

TECHNICAL

Identical to M12 / M13, but with 20,000 additional iterations. The largest preset continue jump. For MCMC training this is often what makes the difference between “okay” and “benchmark-grade”; for Classic, beyond 35–40K experience shows little is added.

IN PLAIN WORDS

Adds 20,000 additional training steps, the maximum continue value. Use this when you really want to squeeze out the last bit of quality. For classic training beyond 40,000 steps this often doesn’t bring much more — whereas for MCMC it’s often worthwhile, because convergence sets in more slowly there. Expect significant extra runtime depending on the scene. Like M12 and M13, this entry is also only available in the full version.

Viewport Menu

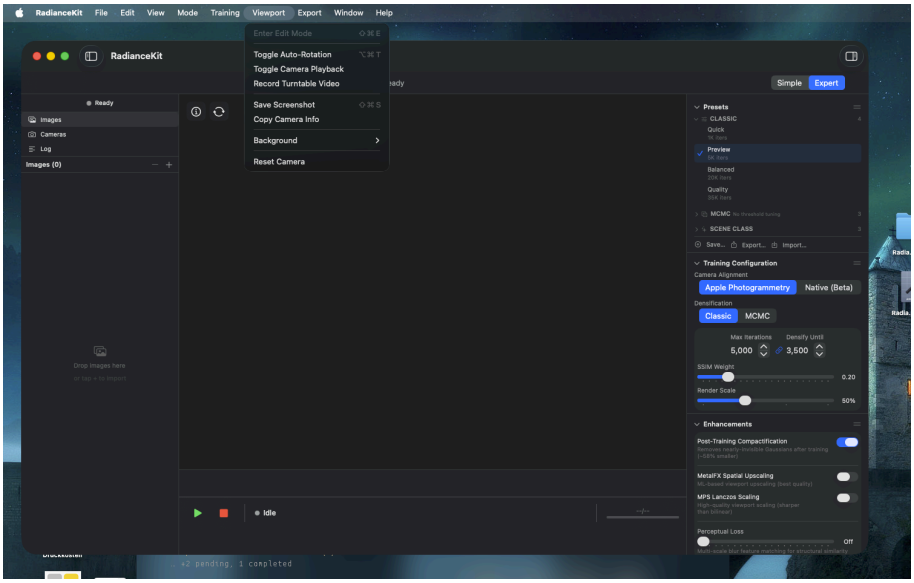

Figure 4: Viewport menu with Edit Mode, camera controls, and Background submenu

Controls the 3D viewport: Edit Mode for Gaussian selection and cleanup, camera controls (auto-rotation, playback, recording), screenshot, background color, and reset.

M15 Viewport > Enter/Exit Edit Mode

WHERE

Menu bar → Viewport → Enter Edit Mode (or “Exit Edit Mode”, depending on state). ⇧⌘E.

TECHNICAL

The entry title is dynamic and shows “Exit Edit Mode” or “Enter Edit Mode” depending on the state. On press, Edit Mode is toggled on the viewport renderer. On leaving Edit Mode, the current selection is also reset. Edit Mode activates click selection on Gaussians, box selection, and the deletion of marked Gaussians (see the editor area of the UI). Disabled as long as no viewport renderer is attached.

IN PLAIN WORDS

Toggles between normal 3D view and an editing mode in which you can mark and delete individual Gaussians (e.g., floaters or outliers in the background). On exit, the selection is automatically reset. The entry stays gray as long as no scene is visible in the viewport yet. The label changes between “Enter Edit Mode” and “Exit Edit Mode” depending on state — so you can always tell which mode you’re currently in.

M16 Viewport > Toggle Auto-Rotation

WHERE

Menu bar → Viewport → Toggle Auto-Rotation (⌘⌥T).

TECHNICAL

Toggles the continuous rotation of the viewport camera around a vertical axis through the scene center on or off. Axis and speed come from the camera control configuration. Auto-rotation is a pure viewport effect and influences neither training nor recording — if you use the turntable video recorder in parallel (M18), auto-rotation delivers exactly the path the recorder captures.

IN PLAIN WORDS

Continuously rotates the camera slowly around your scene so you can see it from all sides without dragging with the mouse. Click again to stop the rotation. Handy when reviewing finished trained scenes or as a background animation for a live demo. If you record a video in parallel (M18), auto-rotation delivers exactly the motion the recorder captures.

M17 Viewport > Toggle Camera Playback

WHERE

Menu bar → Viewport → Toggle Camera Playback.

TECHNICAL

Toggles camera-path playback. If a recorded camera path exists (e.g., from a previous recording or because a `transforms.json` was loaded), the path plays back — the viewport camera no longer moves in response to mouse/trackpad input but reproduces the trajectory frame by frame. Pressing again pauses playback.

IN PLAIN WORDS

Plays back a previously recorded or imported camera path. So you can retrace the original path that was used to capture the scene, or check a planned orbit motion before the video export. While playback is running, mouse and trackpad input is disabled — the camera strictly follows the path. Clicking again pauses playback. If you haven't loaded or recorded a camera path, nothing happens.

M18 Viewport > Record Turntable Video

WHERE

Menu bar → Viewport → Record Turntable Video.

TECHNICAL

Toggles viewport recording. On first press, a frame recording starts to a temporary path; on second press, the recording is finished, encoded, and written to an MP4 path (the path is requested via a save dialog). Unlike Export → Media → Orbit Video (M31), which produces a fixed 360° path at a configurable duration, the turntable recorder captures *live* what you see in the viewport — so you can also record a manual camera move.

IN PLAIN WORDS

Records a video directly in the viewport. Whether the camera rotates automatically or you move it yourself with the mouse — everything you see is saved into an MP4 file. Unlike the "Orbit Video" export (M31), you control the camera move yourself. First click starts the recording, second click ends it and asks where to save. Handy when, e.g., you want to show a particular detail pan that wouldn't be possible with the rigid orbit motion.

M19 Viewport > Save Screenshot

WHERE

Menu bar → Viewport → Save Screenshot (⇧⌘S).

TECHNICAL

Captures a single viewport frame at full render resolution (so not the window pixel layout but the full render-target contents) as a PNG file. The path is requested via a save dialog. The background color (M21–M23) is baked in. MetalFX/MPS upscaling settings from the Enhancements (see I27/I28) take effect if active — the screenshot then shows the upscaled output.

IN PLAIN WORDS

Saves a snapshot of your current 3D view as a PNG image. Handy for marketing material or a quick comparison. Note: The background is part of the image — if you need transparency, export a scene file instead. The resolution corresponds to the internal render target, not your window size — so the image is often sharper than it looks in the window. Any upscaling settings (Inspector → Enhancements) also flow in.

M20 Viewport > Copy Camera Info

WHERE

Menu bar → Viewport → Copy Camera Info.

TECHNICAL

Reads the current viewport camera pose (position, look-at point, up vector) and the FOV values from the camera control and writes them as multi-line text to the clipboard. The format is human-readable (label = value per line), not JSON. Handy for reproducing a specific view for debugging purposes or sharing with support.

IN PLAIN WORDS

Copies the current camera position and viewing direction as text to the clipboard. If you want, e.g., to show a fellow developer where a spot in the scene looks weird, you just paste the text into an email or chat window. The format is human-readable (one line per value), not JSON. Mainly intended for bug reports or support requests.

M21 Viewport > Background > Dark Gray

WHERE

Menu bar → Viewport → Background → Dark Gray.

TECHNICAL

Sets the viewport background color to a dark gray (RGB 0.1/0.1/0.1). The renderer uses this color as the background against which the Gaussians are composited. The default color at app start is controlled by the settings option S3 “Default Viewport Background”.

IN PLAIN WORDS

Tints the background of the 3D viewport dark gray. Standard choice for most scenes — offers good contrast for both bright and dark Gaussians without your eye locking onto a pure black or white area. The color is also baked into screenshots (M19) and orbit videos (M31). If Dark Gray feels too unspectacular, also try Black (M22) or White (M23) for comparison. Which color is active at app start can be set in the settings (S3).

M22 Viewport > Background > Black

WHERE

Menu bar → Viewport → Background → Black.

TECHNICAL

Sets the viewport background color to pure black (RGB 0/0/0). Helps when the scene has many bright floaters and you want to identify them, or for marketing material with a dark look-and-feel.

IN PLAIN WORDS

Black background. Good for very bright scenes or when you want to look into Edit Mode and search for small bright Gaussians (floaters) that get lost in gray. Also ideal for marketing material with a dark, dramatic look. The color is baked into screenshots and orbit videos — if you need transparency for a later composite, black is the worst choice. For dark floaters, switch the other way to White (M23).

M23 Viewport > Background > White

WHERE

Menu bar → Viewport → Background → White.

TECHNICAL

Sets the viewport background color to pure white (RGB 1/1/1). Useful when the scene has predominantly dark content and you want to see dark floaters (typical outdoor background noise).

IN PLAIN WORDS

White background. Handy when the subject shows up better light-on-dark, or to find dark outliers that you can then remove in Edit Mode (M15). For outdoor scenes white is often more useful than black, because typical outdoor floaters tend to be dark. As with the other background options, the color is baked into screenshots and videos.

M24 Viewport > Reset Camera

WHERE

Menu bar → Viewport → Reset Camera.

TECHNICAL

Resets the viewport camera, leaves the training-camera view and stops auto-rotation. That puts the camera back at the initial position (typically: in front of the scene, looking slightly from above), auto-rotation is off, and if the renderer was showing the training camera (one of the SfM poses), it goes back to the free camera.

IN PLAIN WORDS

Brings the viewport camera back to the starting position. If you've gotten lost while moving around or pushed the scene out of frame — click here once and you see again what you should be seeing. At the same time it turns off auto-rotation if it's currently running, and returns from a frozen training camera to the free view. So you get a clean restart of the view in every case.

Export Menu

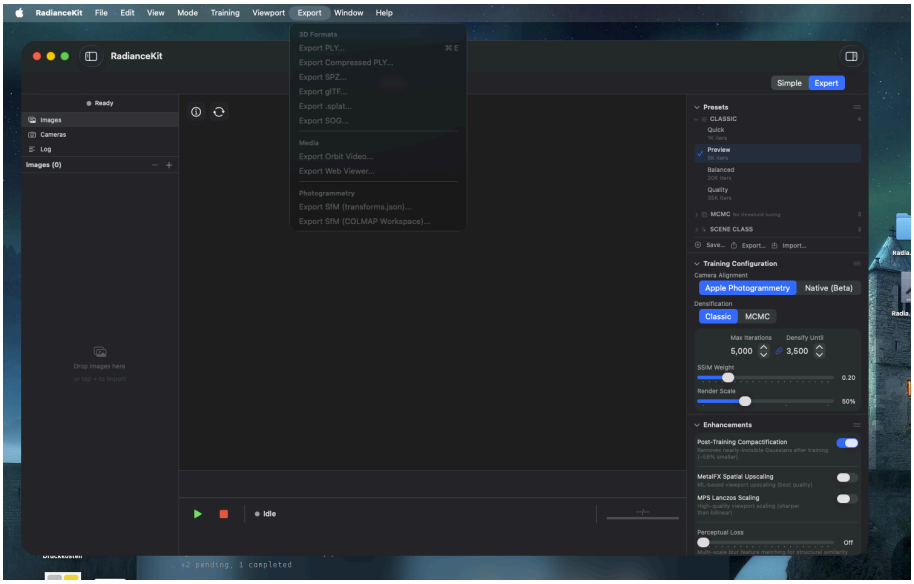

Figure 5: Export menu with three submenu groups — 3D Formats, Media, and Photogrammetry

Eight export targets plus two photogrammetry exports, grouped in three sections (3D Formats, Media, Photogrammetry). The first six are built via a common helper routine that each opens a save dialog and registers the export with the format catalog. The photogrammetry entries have individual logic. All photogrammetry exports and some 3D exports are only available in the full version.

M25 Export > 3D Formats > Export PLY...

WHERE

Menu bar → Export → 3D Formats → PLY (⌘E).

TECHNICAL

Opens a save dialog with default filename `gaussians.ply`. On OK the current Gaussian cloud is written into the standardized ASCII/binary PLY format — compatible with SuperSplat, PolyCam, PlayCanvas and all common 3DGS viewers. Full SH coefficients, full precision (Float32 per field). File size often several hundred MB at ≥ 500K Gaussians.

IN PLAIN WORDS

Saves your 3D scene as a standard PLY file. This is the most universal format — almost every software can load it, from SuperSplat to PolyCam to PlayCanvas. The files do get large, though, often several hundred megabytes. Use PLY when you want to keep working in full quality or archive. If you want to share the scene over the web, look at SPZ (M27) or Compressed PLY (M26) instead — those are much smaller.

M26 Export > 3D Formats > Export Compressed PLY...

WHERE

Menu bar → Export → 3D Formats → Compressed PLY.

TECHNICAL

Writes the Gaussian cloud in the compressed-PLY format with custom quantization of the position, scale, rotation, and SH fields. 5–10× smaller files than the uncompressed PLY (M25) with minimal visual loss. Compatible with SuperSplat (which reads the compressed-PLY standard) and PlayCanvas. Default filename `gaussians_compressed.ply`.

IN PLAIN WORDS

Like normal PLY, but 5–10 times smaller. Quality stays almost the same. Use this when you want to share the file online or send it by email. Works directly with SuperSplat and PlayCanvas. If your target system needs even smaller files (mobile, browser demos), use SPZ (M27) instead — that's compressed even more aggressively. For full editing quality, use the uncompressed PLY (M25).

M27 Export > 3D Formats > Export SPZ...

WHERE

Menu bar → Export → 3D Formats → SPZ.

TECHNICAL

Writes the Gaussian cloud in the SPZ format — the compressed splat format published by Niantic with aggressive quantization (~90 % smaller than uncompressed PLY). Mainly optimized for web viewers and mobile apps. Compatible with Niantic Splatt3R, gsplat.js, and the Niantic browser viewer.

IN PLAIN WORDS

One of the smallest formats. About 10× smaller than a normal PLY. Use this primarily when you want to show the scene in a browser or view it on a phone app. For maximum quality, PLY is the better choice. SPZ was developed by Niantic and works directly with gsplat.js, Splatt3R, and the Niantic web viewer. Because of the strong compression, you can no longer easily keep training SPZ files — use PLY for editing.

M28 Export > 3D Formats > Export glTF...

WHERE

Menu bar → Export → 3D Formats → glTF.

TECHNICAL

Writes a `.glb` file (binary glTF) with the KHR_gaussian_splatting extension. Standard-conformant, suitable for pipelines that use glTF engines like Babylon.js or Three.js and implement the KHR_gaussian_splatting extension.

IN PLAIN WORDS

Saves the scene in the glTF format that many 3D programs and web engines understand — provided they support the Gaussian-Splatting extension. If you have a specific 3D pipeline (e.g., Three.js or Babylon.js) that understands it, this is your format. The file comes out as binary `.glb` — a single package that contains everything. For classic splatting workflows PLY or SPZ is usually the better choice, since more tools understand those directly.

M29 Export > 3D Formats > Export .splat…

WHERE

Menu bar → Export → 3D Formats → .splat.

TECHNICAL

Writes the Antimatter15 `.splat` format — fixed-size 32 bytes per Gaussian (position as 3× Float32, scale as 3× Float32, rotation as 4× Uint8 normalized quaternion, RGB+opacity as 4× Uint8). No SH coefficients higher than DC. Smallest file with direct browser compatibility. For gsplat.js and antimatter15's online demo viewer.

IN PLAIN WORDS

The simplest web-viewer format. Small and immediately displayable in any browser. But loses detail lighting (higher SH coefficients are lost — the splat looks the same from every angle instead of reacting to the light). Good for maximum web performance, but for photorealism, SPZ or PLY is better. Works with the antimatter15 online viewer and gsplat.js. Each Gaussian takes a fixed 32 bytes, which makes the format simple and compatible — but at the cost of detail depth.

M30 Export > 3D Formats > Export SOG…

WHERE

Menu bar → Export → 3D Formats → SOG.

TECHNICAL

Writes the Gaussian cloud in the SOG format. SOG ("Self-Organizing Gaussian") is the PlayCanvas format with texture-atlas layout and WebP compression of the quantized data. Scales with 15–20× better size ratio than PLY. The export calls `cwebp` as an external tool internally — so in the sandbox variant (App Store) potentially restricted.

IN PLAIN WORDS

Very small format for PlayCanvas workflows. About 15–20 times smaller than PLY because the data is packed into a texture-atlas layout and WebP-compressed. If you don't have a PlayCanvas workflow, SPZ or Compressed PLY is usually the better choice. The export calls `cwebp` as an external tool internally — in the App Store version (sandbox), this step can be restricted.

M31 Export > Media > Export Orbit Video...

WHERE

Menu bar → Export → Media → Orbit Video.

TECHNICAL

Renders a 360° orbit around the scene center and encodes it as MP4 (H.264) or MOV (HEVC, depending on system default). Unlike M18 (live recording), the path here is fixed — duration is chosen in the settings or in the Simple Mode export step.

IN PLAIN WORDS

Automatically produces a turnaround video of your scene. No manual moving needed. Good for social media or a quick demo. If you want to control the camera yourself, use Record Turntable Video (M18) instead. The path is fixed: a full 360° orbit around the scene center; you choose duration in the settings or in the Simple Mode export step. The video is output as H.264 MP4 or HEVC MOV depending on the system.

M32 Export > Media > Export Web Viewer...

WHERE

Menu bar → Export → Media → Web Viewer.

TECHNICAL

Packages a standalone HTML viewer (gsplat.js-based) plus the Gaussian data Base64-encoded into a single `.html` file. This file runs offline in any modern browser — no server dependencies, no external URLs. File size is about a factor of 1.3 larger than the SPZ variant (because of Base64 overhead).

IN PLAIN WORDS

Saves your scene as a self-starting web page. Double-click on the HTML file → browser opens → finished interactive 3D scene. Works without internet, can be sent by mail, is the simplest way to share the result with friends or clients. The file contains the complete gsplat.js viewer and the Gaussian data in a single document — nothing is loaded from the web. File size is about a third larger than an SPZ export, but in exchange the recipient needs no additional software.

M33 Export > Photogrammetry > Export SfM (transforms.json)…

WHERE

Menu bar → Export → Photogrammetry → Export SfM (transforms.json).

TECHNICAL

Dedicated export path (not via the common helper routine), because no Gaussian cloud but the SfM result is being exported. Opens a save dialog with `transforms.json` as the default and content type `json`. On OK a nerfstudio-compatible `transforms.json` with camera intrinsics, poses (as a 4×4 matrix in NeRF convention) and frame paths is written. Help text in the UI points out that the training images must be copied alongside as a sibling folder `images/`. Active only if an SfM result exists and the full version is unlocked.

IN PLAIN WORDS

If you want to use the SfM result in another piece of software like nerfstudio, Brush, gsplat, or OpenSplat, this is where you export the camera positions. Place your training images additionally into an `images/` folder next to the `transforms.json` file — otherwise the target program cannot match the images. The entry is grayed out as long as no SfM result exists, and locked in the free trial version. For the COLMAP-workspace workflow, use M34 instead.

M34 Export > Photogrammetry > Export SfM (COLMAP Workspace)…

WHERE

Menu bar → Export → Photogrammetry → Export SfM (COLMAP Workspace).

TECHNICAL

Opens a save dialog with default name `colmap-workspace` (without extension, because it's a folder). Writes a standard COLMAP workspace with `sparse/0/cameras.bin`, `images.bin`, `points3D.bin`. Lets you open an SfM reconstruction computed or imported in RadianceKit in other tools like Postshot, Nerfstudio, or Meshroom, or for an A/B re-run reload it as already-computed input in RadianceKit itself (via M5) — saves compute time. Active only if an SfM result exists and the full version is unlocked.

IN PLAIN WORDS

Like M33 but in COLMAP format instead of nerfstudio. If you use Postshot, Meshroom, Nerfstudio, or another tool with a COLMAP workflow, this is your export. A handy side effect: You can later reload this folder via M5 in RadianceKit and save the SfM compute time on the next run — especially on large scenes a time saving of hours. Like M33, only available when an SfM result exists, and locked in the free trial version.

Help Menu

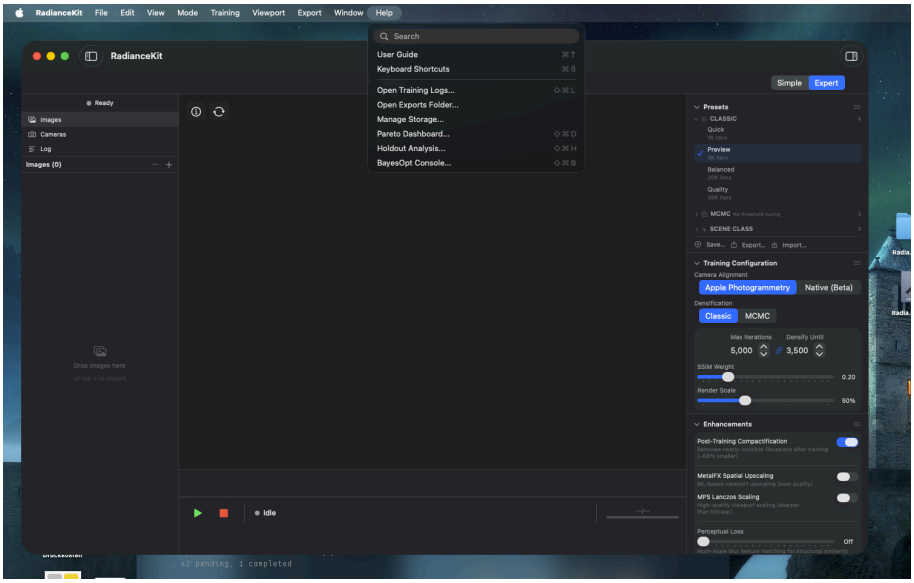

Figure 6: Help menu with documentation, folder, and analysis entries

Seven entries: two documentation windows (User Guide, Keyboard Shortcuts), three folder shortcuts (Training Logs, Exports, Storage), and three analysis windows (Pareto Dashboard, Holdout Analysis, BayesOpt Console). Apple-style, the Help menu appears all the way right. The standard Help menu is completely replaced by RadianceKit's own variant.

M35 Help > User Guide

WHERE

Menu bar → Help → User Guide (⌘?).

TECHNICAL

Opens the User Guide window. It shows navigation with a topic sidebar and a scrollable detail area at default size 860×640. The contents are statically stored (not parsed from Markdown).

IN PLAIN WORDS

Opens the in-app guide. If you don't want to read everything in this manual, you'll find the key steps directly in the program. The guide is built as its own window with a topic sidebar — so you can jump directly to individual topics. The contents are shorter than this manual and focus on the most common workflows.

M36 Help > Keyboard Shortcuts

WHERE

Menu bar → Help → Keyboard Shortcuts (⌘/).

TECHNICAL

Opens the Keyboard Shortcuts window — a simple scroll layout with all app shortcuts, grouped by top-level menu. Default size 440×560. Contents are likewise statically stored.

IN PLAIN WORDS

Opens a window with the complete list of all keyboard shortcuts. If you can't remember, e.g., which key starts training, look there. A summary is also at the end of this chapter. The list is grouped by top-level menu, so you jump to the right area quickly. Helpful when you're switching from a mouse style to a keyboard style.

M37 Help > Open Training Logs…

WHERE

Menu bar → Help → Open Training Logs… (⇧⌘L).

TECHNICAL

Computes the log folder as `~/Documents/RadianceKit/Logs`, creates it if necessary, and opens it in Finder. Each training run writes its own JSONL file `training_YYYY-MM-DD_HHmmss.jsonl` there.

IN PLAIN WORDS

Opens the folder with all previous training protocols in Finder. If something went wrong or you want to look up exactly when training converged to which value, you'll find that here in JSONL files. Per training run exactly one file with a timestamp is created — you can also feed it into other tools or email it to support. If you want a graphical analysis, the Pareto Dashboard (M40) is the better starting point.

M38 Help > Open Exports Folder...

WHERE

Menu bar → Help → Open Exports Folder...

TECHNICAL

Analogous to M37 but with `~/Documents/RadianceKit/Exports`. Created on the first auto-test run or on the first click; after that the default paths of all auto-test exports (e.g., `autotest_<timestamp>.ply`) land there. Exports selected manually via the save dialog do NOT necessarily go here but wherever the user saves them — so this folder is mainly of interest for auto tests.

IN PLAIN WORDS

Opens the folder where the app places its own exports (especially auto-test runs). If you manually placed an export somewhere else with the save dialog, it's there and not in this folder. Handy for cleaning up or for checking how much space previous test exports take up. If you want a complete overview including logs and scene bundles, use Manage Storage (M39) instead.

M39 Help > Manage Storage...

WHERE

Menu bar → Help → Manage Storage...

TECHNICAL

Opens the storage browser (see Chapter 4 Auxiliary Windows, IDs W7–W12). Lists all persisted scenes, training logs, exports, and caches in the `~/Documents/RadianceKit/` folder with size, allows reveal in Finder and move to trash per entry.

IN PLAIN WORDS

Opens a window browser that shows you how much disk space RadianceKit takes up — per scene, log, and export. You can delete individual items directly, without going into Finder. Handy after extended use, when the disk is filling up — previous logs and auto-test exports can add up to several gigabytes. Via reveal in Finder you can always get to the classic view as well.

M40 Help > Pareto Dashboard...

WHERE

Menu bar → Help → Pareto Dashboard... (⇧⌘D).

TECHNICAL

Opens the Pareto Dashboard (see Chapter 4, IDs W13–W22). The dashboard loads all JSONL training logs from `~/Documents/RadianceKit/Logs/`, organizes them by scene and preset, and draws a Pareto scatter plot (default: loss vs Gaussians, optionally loss vs wallclock or PSNR vs iterations).

IN PLAIN WORDS

Opens an overview of all previous training runs as a chart. You see at once which run delivered the best balance of quality and size. Handy when you want to compare different presets. By default, the chart shows loss against Gaussian count — but you can also switch to wallclock time or PSNR. The data comes from the JSONL training logs (M37); the more runs you have, the more meaningful the analysis.

M41 Help > Holdout Analysis...

WHERE

Menu bar → Help → Holdout Analysis... (⇧⌘H).

TECHNICAL

Opens the Holdout Analysis window (see Chapter 4, IDs W23–W29). Loads a `transforms.json`, draws the cameras as a 3D globe, and allows train/test fold splits (angular or linear, 2–8 folds). Output is a `fold-assignment.json` that training can use as the test set in the respective training configs.

IN PLAIN WORDS

Helps you split your camera captures into training and test sets — so you can objectively measure how good your scene is (on images training never saw). More of a research and benchmark tool. The cameras are shown as a 3D globe; you can choose between 2 and 8 folds, either evenly in angle or linearly across order. The result is a small JSON file that training then uses as the test set.

M42 Help > BayesOpt Console…

WHERE

Menu bar → Help → BayesOpt Console… (⇧⌘B).

TECHNICAL

Opens the BayesOpt Console (see Chapter 4, IDs W30–W39). Loads predefined search spaces (e.g., “MCMC scale-reg + opacity-reg + ssim”), runs Bayesian-optimization trials asynchronously, and shows the convergence curve and trial log live.

IN PLAIN WORDS

A built-in auto-tuner console. Instead of manually trying out different parameters, the app can run by itself overnight and at the end suggest the best values for your scene. Very advanced tool — for most workflows a good preset (see Chapter 7) is sufficient. You choose a predefined search space (e.g., “MCMC scale-reg + opacity-reg + ssim”) and see the convergence curve live as well as the trial log. Plan for several hours to days depending on setup.

Note: Cmd-Z in the Edit Menu

Since May 2026, the Project Navigator in Expert Mode supports deleting imported images via the minus button or Backspace key, and undoing via `Cmd-Z`. This Cmd-Z action appears in the macOS Edit menu (provided by SwiftUI) as “Undo Remove Image” as long as a deleted image is still recoverable. It is registered via the standard-conformant system, not in ; therefore there is no dedicated M-ID entry in the inventory.

Keyboard Shortcuts Overview

Menu entry	Shortcut
File > Open Scene…	⌘O
File > Save Scene…	⌘S
File > Import COLMAP / Metashape Workspace…	⇧⌘I
File > New Project	⇧⌘N
Mode > Simple Mode	⌘1
Mode > Expert Mode	⌘2
Training > Start Training	⇧⌘T
Viewport > Enter/Exit Edit Mode	⇧⌘E
Viewport > Toggle Auto-Rotation	⌘⌥T
Viewport > Save Screenshot	⇧⌘S
Export > 3D Formats > PLY	⌘E
Help > User Guide	⌘?
Help > Keyboard Shortcuts	⌘/
Help > Open Training Logs…	⇧⌘L
Help > Pareto Dashboard…	⇧⌘D
Help > Holdout Analysis…	⇧⌘H
Help > BayesOpt Console…	⇧⌘B

Edit menu (system-provided, in Expert Mode with active Project Navigator selection):

Action	Shortcut
Undo Remove Image	⌘Z
Remove Selected Image	Backspace / Delete

CHAPTER

Chapter 2 — Inspector (Expert View)

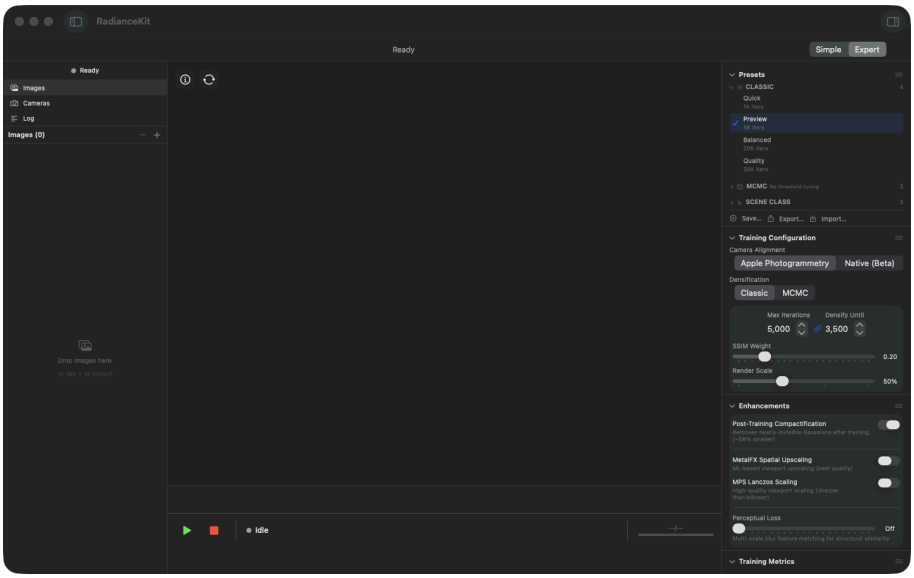

Figure 7: Empty Expert Mode — Project Navigator on the left (Images 0, Cameras, Log), empty viewport in the middle, Inspector on the right with Presets/Training Configuration/ Enhancements/Training Metrics sections

Empty Inspector before import: The left sidebar shows the Images counter 0 and the drop hint “Drop images here / or tap + to import”. The Inspector on the right is fully functional, but presets are only informative (no active training). The default preset “Preview” (5K iters) is marked. Camera Alignment is set to Apple Photogrammetry, Densification to Classic, SSIM Weight to 0.20, Render Scale to 50 %. Empty states are shown in Training Metrics (“Start training to see live metrics”) and Loss History (“Loss curve will appear during training”).

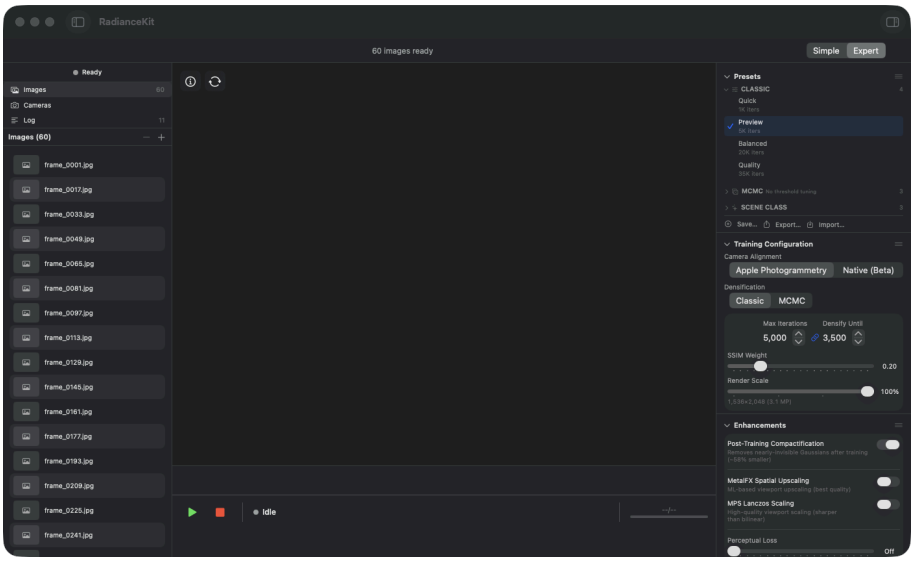

Figure 8: Inspector with 60 flowers images loaded — the image sidebar shows the first filenames frame_0001.jpg ff, header reads "60 images ready"

Inspector after import: Header status reads "60 images ready". The image sidebar lists all 60 imported frames (`frame_0001.jpg` to `frame_0945.jpg` , every 16th frame of the 960-cam Bouquet dataset as a subset for quick iterations). The auto render scale logic checks the image resolution (1536×2048 = 3.1 MP) and adjusts Render Scale accordingly. The Play button (green, bottom left) is now active and starts training with the active preset.

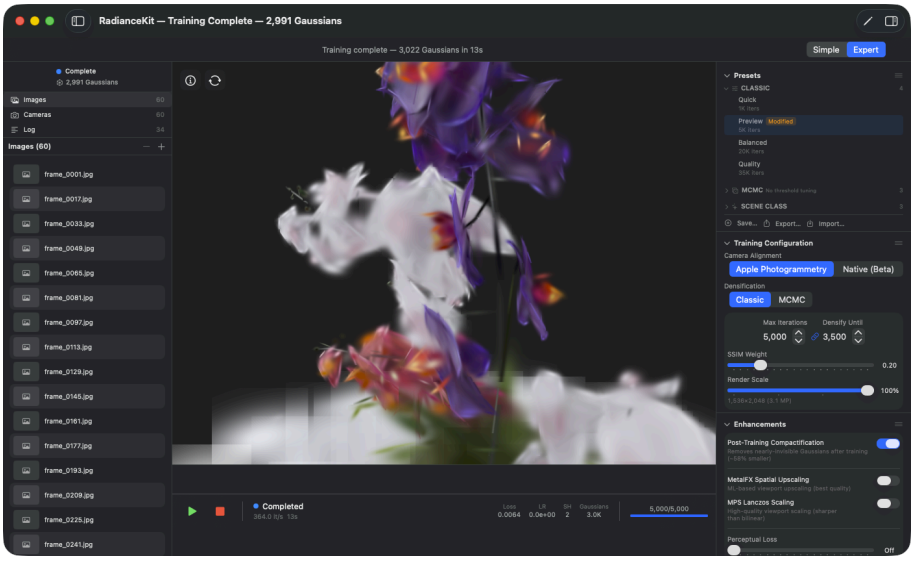

Figure 9: Inspector mid-training — the live viewport shows the flowers Bouquet reconstruction, metrics bar at the bottom (Loss / LR / Gaussian count / iterations), preset card "Preview" with "Modified" badge if parameters were adjusted

Inspector during training: The title bar shows global progress "RadianceKit — Training NN %". The viewport renders the running Gaussian reconstruction in real time (updated every 50 iterations — live preview interval can be set in Settings → General → Training → Live Preview). Metrics bar below the viewport: current Loss, Learning Rate, Gaussian count, and iteration counter (e.g. 1,600/5,000 with the Preview preset). The Inspector

preset card “Preview” gets a “Modified” badge as soon as any parameter deviates from the built-in default. The “Log” sidebar collects SfM and training stage events.

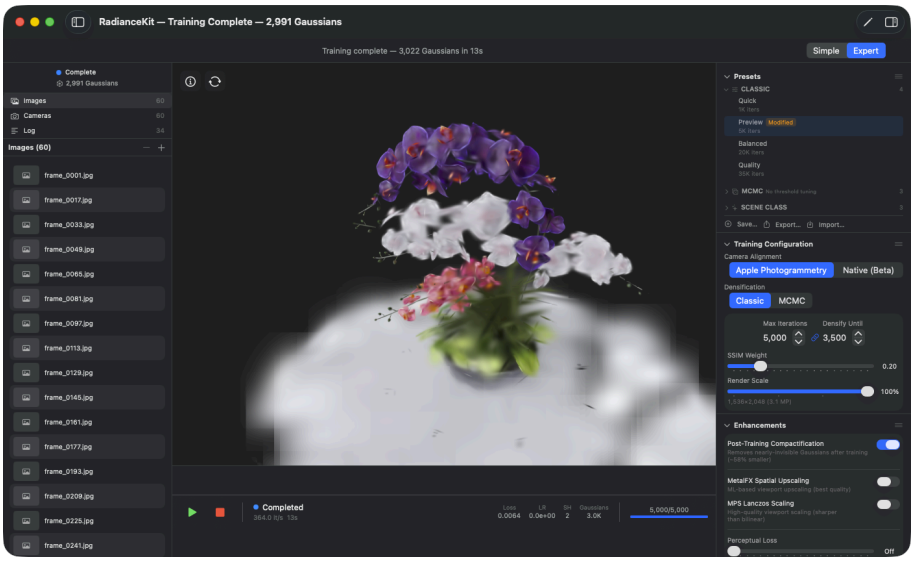

Figure 10: Inspector after training completes — the viewport shows the finished flowers Bouquet reconstruction (2,991 Gaussians after 5K iterations in 13s), title bar “Training Complete — 2,991 Gaussians”

Inspector after training: The title bar shows the final Gaussian count (here 2,991 — very compact, because the synthetic Blender Bouquet scene has simple geometry on a bright background). The viewport shows the finished point cloud — orbital drag navigation is active (rotates around the scene center). The Training Metrics section is now filled with final values, the Loss History chart shows the course of the entire 5,000 iterations. The Export section at the bottom is now active (all format buttons enabled).

The Inspector is the right sidebar in Expert Mode (⌘2). It bundles all training-relevant parameters into seven collapsible sections. The default top-to-bottom order on first launch is: Look, Presets, Training Configuration, Metrics, Loss Chart, Enhancements, and Export. The “Look” section (post-training image adjustments) is the real UI rename of the former “Finishing” section — its internal enum `rawValue` stays “Finishing” for persistence reasons, the displayed heading reads “Look”. Each section can be collapsed by clicking the header, and the order can be rearranged via drag-and-drop (`InspectorView.swift:81–97`). **On first launch all seven sections start collapsed** (`inspectorCollapsedSections` defaults to `Set(InspectorSection.allCases)`); the app state persists collapse and ordering preferences afterwards across app starts.

A number of controls from the Inspector also appear in nearly identical form in the Settings (Chapter 3) — typically SfM backend, sky masking, and similar defaults. The split is deliberate: Settings provides the app-global template for newly created projects, the Inspector overrides those values for the currently open project. Once you know the operating logic of one side, you can use the other blindly.

The left column in Expert Mode — the Project Navigator — is not part of the Inspector, but it’s its direct neighbor. There you can select imported images by clicking, preview them with Spacebar via Quick Look, and delete them via the minus button or the Delete key (with Cmd-Z to undo). The Inspector follows the current sidebar selection with context-specific detail information, but the seven main sections always stay available.

Look section (L1–L5)

The Look section (internal `rawValue` still “Finishing”) is the topmost Inspector section and collects the **post-training** image adjustments in one place. All sliders work **non-destructively**: each slider re-applies the `FinishingPass` to an unchanged pristine snapshot (original DC color, opacity, scaling) — the adjustment is therefore **idempotent**, not cumulative. The result appears **live in the viewport** (WYSIWYG, exactly like the later export) and is **baked into every export**. The section is only available **after a training run completes** (before that it reads “Available after a training run completes.”); its values are **reset on each new training**. While an export is running, all sliders are **locked** — a lock hint “Locked while exporting — the file uses the current settings.” appears and the GroupBox is disabled.

L1 Saturation slider

WHERE

Inspector → Look section → GroupBox → Saturation.

TECHNICAL

Slider 0.5–1.2, displayed with two decimals (e.g. “1.00”). Scales the SH-DC chroma of each Splat by its luminance value: 1.0 = unchanged, < 1.0 = desaturated (color pulled toward grayscale), > 1.0 = punchier. Mathematically the DC color is recomputed from the pristine snapshot (`desaturateDC`), so repeated dragging doesn’t accumulate. Was validated on DJI drone footage (Pensford viaduct), which tends to oversaturate — the drone default is 0.82. Affects only the color base (SH degree 0), higher SH coefficients stay untouched.

IN PLAIN WORDS

How vivid the colors of the finished Splat are. 1.00 leaves everything as trained, lower values pull the color toward gray — good for drone or video footage that often comes out oversaturated. Values above 1.0 make it punchier. You can slide back and forth as much as you like without anything “building up”, because the app always recomputes from the unchanged original state. Visible live in the viewport and exactly so in the export.

L2 Splat length slider

WHERE

Inspector → Look section → GroupBox → Splat length.

TECHNICAL

Slider 0.3–1.0, displayed with two decimals. Pulls the three scale axes of each gaussian in log-space toward their mean (`shortenScale` , factor `alpha`): 1.0 = unchanged, smaller values make elongated "needle" Splats rounder, 0 would be pure spheres. Targets needle-like, over-stretched Splats without changing the overall size, and thereby reduces typical "confetti" artifacts. Applied from the pristine snapshot (original log-scaling), hence idempotent. Commutes with Splat size (L3), because both work in log-space.

IN PLAIN WORDS

Makes over-long, splintery Splats rounder. 1.00 leaves the shape as trained, lower values squash the elongated "needles" into rounder blobs — that calms grainy reconstructions plagued by confetti artifacts. The overall size stays the same, it's only about the elongation. Can be combined safely with Splat size (L3).

L3 Splat size slider

WHERE

Inspector → Look section → GroupBox → Splat size.

TECHNICAL

Slider 0.5–2.0, displayed with two decimals. Scales each gaussian uniformly on **all** three axes (`sizeScale`): 1.0 = unchanged, < 1.0 = smaller/denser/sharper, > 1.0 = bigger/"fluffier" (fills gaps between the Splats). Since the scalings live in log-space, the multiplication is realized as an additive `log(factor)` offset — that commutes with Splat length (L2), because a constant offset leaves the deviation-from-mean untouched. From the pristine snapshot, so idempotent. New in this version.

IN PLAIN WORDS

Scales all Splats uniformly larger or smaller. 1.00 is the trained state, lower values make the point cloud tighter and sharper, higher values cover gaps between the Splats (looks softer/"fluffier"). Handy to visually close a holey reconstruction or, the other way around, to reveal more detail. Gets along with Splat length (L2) without trouble — the two sliders don't influence each other.

L4 Fade far region (with sub-sliders)

WHERE

Inspector → Look section → GroupBox → Toggle “Fade far region” plus the sub-sliders “Fade start ×radius” and “Fade floor”.

TECHNICAL

Toggle that activates a radial opacity falloff with the distance from the camera centroid — the weakly observed “far confetti” in the background gets faded out. **Orbit-only**: the toggle is disabled when `finishingContext.fadeEligible` is false (linear flights, too few or degenerate cameras); then, instead of the sub-sliders, the hint “Far-fade applies only to orbit captures (not this scene).” appears. Eligibility is determined via the azimuth coverage of the camera positions (an orbit circles the centroid and fills many compass sectors, a linear flight only ~2). Two sub-sliders control the geometry: **Fade start ×radius** (1.0–3.0) sets the inner radius as a multiple of the orbit radius, within which full opacity applies; **Fade floor** (0.0–1.0) is the opacity factor far beyond the fade radius. Important: the fade **skips the sky-dome region** (the frozen gaussians of indices [0, frozenCount)), so the intentional background dome isn’t dimmed along with it.

IN PLAIN WORDS

Fades out the mushy remnants at the outer scene edge — exactly the “far confetti” blobs that float far back in all-around captures. Only works on real orbit/encircling captures; with straight drone flights or too few cameras the switch is grayed out and a hint explains why. When it’s active, two fine controls join: “Fade start ×radius” sets from which distance (as a multiple of the orbit radius) the fading begins, “Fade floor” how strongly the far Splats still stay visible at the end (0 = gone completely, 1 = unchanged). A deliberately reconstructed sky dome (I44) is never touched in the process — the sky stays intact.

L5 Reset finishing button

WHERE

Inspector → Look section → GroupBox → “Reset finishing” (at the bottom, small button).

TECHNICAL

Resets all Look settings to the defaults (`FinishingPass.Settings()` = Saturation 1.0, Fade off, Splat length 1.0, Splat size 1.0) and immediately triggers a fresh finishing pass, so the viewport snaps back to the unchanged trained state. `controlSize(.small)`. Since the whole Look stack computes idempotently from the pristine snapshot, “back to default” is exactly the original training output — no quality loss from repeated back-and-forth. Like all controls of the section, locked during a running export.

IN PLAIN WORDS

Resets all Look sliders to default with one click (Saturation 1.00, Fade off, both Splat sliders to 1.00) — the viewport then shows exactly the freshly trained result again. Handy when you’ve played around and want to start cleanly from scratch. Because the app always computes from the original state, there’s no quality loss in the process. While an export is running, the button (like the sliders) is locked.

Presets section (I1–I11)

The Presets section is the fastest way to apply a tested configuration. Built-in presets (Capture Class, Classic, MCMC, Hybrid) provide reproducible starting points from 560+ documented experiments; your own presets can be saved, exported, imported, and shared. The list is grouped by categories (Capture Class, Classic, MCMC, Hybrid, Custom) and more than one category can be expanded at the same time. The context menu (right-click on a row) makes Export, Duplicate, and — for custom presets — Delete reachable.

I1 Save… button

WHERE

Inspector → Presets section → Save… button (action bar at the bottom).

TECHNICAL

Opens a popover with text field and Save/Cancel buttons. The current TrainingConfig state is persisted as a new custom preset (JSON-encoded, stored app-wide). The save process copies all 81 training parameters plus the current densification strategy. The preset lands automatically in the Custom category, regardless of which built-in preset it was derived from. Empty names and pure whitespace input are rejected. Already existing names are not rejected — every preset has its own internal ID, duplicate names are technically allowed but practically confusing.

IN PLAIN WORDS

Saves your current configuration as a reusable preset. Click the button, type a name in the popover, and click Save — all 81 parameters including densification strategy land under the chosen name in the Custom category. You need this when you've put in effort and don't want to fiddle from scratch on your next project. Especially handy for recurring setups like "Drone 4K" or "Indoor fast". Duplicate names are technically allowed but practically confusing — better pick something descriptive.

I2 Preset Name text field

WHERE

Save popover → "Preset Name" text field.

TECHNICAL

Simple text field with rounded border, wide shape. The value is taken as the preset name when you click the Save button. No length limit in the UI, but the stored name must be JSON-encodable and displayable in the UI lists — emoji and umlauts work. The content is automatically reset to an empty string when the popover opens. The Save button stays disabled as long as the field is empty after trim. There is no auto-suggest and no pre-fill with the name of the currently active preset.

IN PLAIN WORDS

This is where you type the name for your preset. Pick something descriptive like "Drone 4K 30fps" or "Indoor fast" — that'll help you find it again later in the Custom category. Emoji and umlauts are allowed, there's no hard length limit. As long as the field is empty or contains only spaces, the Save button stays grayed out. When you reopen the popover, the field is empty again — there's no pre-fill with the active preset's name.

I3 Cancel button (Save dialog)

WHERE

Save popover → Cancel button (left).

TECHNICAL

Closes the popover without saving. Discards the text field content — on next open it will be reset to empty by the Save… button logic (I1). Standard button style, no confirmation dialogs, no hotkeys. The current TrainingConfig stays unchanged, since the save path wasn't even executed.

IN PLAIN WORDS

Closes the Save popover without saving anything. If you've changed your mind, mistyped, or accidentally opened the dialog — just click Cancel. Your current training configuration stays unchanged, because nothing was written. When you reopen the popover, the name field starts empty again. No safety prompt, no hotkey — just click and gone.

I4 Save button (Save dialog)

WHERE

Save popover → Save button (right, prominent style).

TECHNICAL

Triggers the actual persistence. Validates again that the name is non-empty (defensive check) and then writes the current TrainingConfig as JSON to app storage. Then closes the popover. Highlighted blue, grayed out while the text field is empty. If saving fails (e.g. because app storage is full — very unlikely), there's currently no visible error dialog; the preset would simply not appear at the next app start.

IN PLAIN WORDS

With a click on Save you take over the name and write your current setup as a new preset. The popover closes, the preset immediately appears in the Custom category of the preset list and can from then on be activated by click. The button is highlighted blue (`borderedProminent`) and stays grayed out as long as the name field is empty. If saving fails (e.g. UserDefaults full), there's no visible error dialog — the preset would simply be missing at the next app start.

I5 Export… button

WHERE

Inspector → Presets section → action bar → Export… button.

TECHNICAL

Exports the currently selected preset as a `.radiancepreset` file (internally JSON). Disabled if no preset is selected. On click the app opens a save dialog with a preset filename (preset name + `.radiancepreset` extension). The saved format contains the complete TrainingConfig plus metadata (name, category, ID, built-in flag). Double-clicking in the Finder opens the app — but **not** automatically the import; the user has to use the Import button (I6).

IN PLAIN WORDS

Select a preset in the list and click Export — then you can save it as a `.radiancepreset` file and e.g. send it to a colleague or transfer to a second Mac. The recipient loads it on their end with the Import… button (I6) again. Works equally well for built-ins and your own custom presets. The button is grayed out as long as nothing is clicked in the list. Tip: via the context menu (I8) it's even faster — there you don't have to select the preset first.

I6 Import… button

WHERE

Inspector → Presets section → action bar → Import… button.

TECHNICAL

Opens a file dialog that only allows `.radiancepreset` files (multiselect disabled). On selection the JSON file is loaded, validated, and inserted into the Custom category — with a new internal ID so no collisions with built-ins occur. The import automatically sets the category to Custom, even if the exported preset originally was e.g. a built-in. Corrupted or files incompatible with an older schema version are silently rejected, without error dialog (console log does provide info).

IN PLAIN WORDS

Reads a `.radiancepreset` file from disk. Useful when someone sends you a proven setup or you yourself want to keep your favorite presets in sync across multiple Macs. Imported presets always land in the Custom category — even if they were originally exported from the built-ins. Corrupted or outdated files are silently ignored; the console log shows the reason. Multiselect is disabled in the dialog, so only one file per click.

I7 Preset row (click activation)

WHERE

Inspector → Presets section → every preset row in every category.

TECHNICAL

Clicking a preset row replaces all fields of the TrainingConfig with the values from the preset, remembers the ID of the active preset, and resets the Modified status. The active checkmark in front of the row only appears when the preset is selected AND unmodified. As soon as a value in TrainingConfig is changed (slider, stepper, toggle in the other Inspector sections), an orange "Modified" badge appears after the name. Built-in presets cannot be overwritten — on modification you have to create a copy via the Save button (I1).

IN PLAIN WORDS

Clicking a row activates the preset and takes over all values stored there into the current training settings. The checkmark in front of the name shows which preset is currently active. As soon as you afterwards adjust any slider, stepper, or toggle in the other sections, an orange "Modified" badge appears after the name — because your setup now deviates from the preset. Built-in presets cannot be overwritten; if you want to keep changes, create a copy via the Save… button (I1) or duplicate the preset (I9).

I8 Context menu "Export…"

WHERE

Right-click on any preset row → first entry "Export…".

TECHNICAL

Identical functionality as I5 (Export… button), but more conveniently reachable — without the preset having to be selected first. Exports directly the preset clicked in the row. Works for all preset categories alike (built-in or custom), no restriction. The export contains the built-in flag and the original category, but on re-import the category is mapped to Custom as described under I6.

IN PLAIN WORDS

Quick way to export — right-click on the desired preset and choose "Export…". Saves the detour via pre-clicking and then pressing the Export… button. Works for all categories alike, also for built-ins. The created `.radiancepreset` file is identical to the one from I5; on later re-import it automatically lands in the Custom category.

I9 Context menu “Duplicate”

WHERE

Right-click on any preset row → second entry “Duplicate”.

TECHNICAL

Clones the preset into the Custom category. Creates a new internal ID, appends “ Copy” to the name, and saves the copy. Also works for built-in presets — the clone is then editable. The original stays untouched. The TrainingConfig is copied value-by-value (JSON round-trip), so no reference bindings between original and copy exist.

IN PLAIN WORDS

Creates an editable copy of a preset in the Custom category. Handy if you e.g. want the built-in “Quality” preset as a starting point and then only want to nudge the SSIM slider a bit. Workflow: duplicate, rename (context menu or new Save… run), adjust, done. The original stays untouched — you can return to it any time. Also works for built-ins, which is the only way to use their values as a basis and at the same time make them editable.

I10 Context menu “Delete”

WHERE

Right-click on your own preset rows → last entry “Delete” (red, destructive).

TECHNICAL

Only visible for custom presets. Built-ins cannot be deleted. The entry is marked as destructive, appears red in the context menu, and is set off behind a divider so you don’t click it by accident. There is **no** confirmation dialog — a click deletes the preset immediately. The deleted preset is not recoverable (Cmd-Z does not work here — Undo exists in the current build only for the image list, not for preset operations). If the deleted preset was just active, the current TrainingConfig stays unchanged, only the active preset selection is nulled.

IN PLAIN WORDS

Delete your own presets. For the built-ins (Quick, Preview, Balanced, Quality, Ultra Detail, Drone / Aerial, 360° Walkaround, Photo / Object, etc.) “Delete” isn’t even visible — you can’t kill them by accident. Heads up: there’s no safety prompt and no undo, one click and the preset is gone. If you’re not sure, first pull a backup to disk via Export… (I5/I8) — you can re-import it any time. If the preset was just active, your TrainingConfig stays unchanged, only the checkmark disappears.

I11 Category header (expand/collapse)

WHERE

Inspector → Presets section → every category header (Capture Class, Classic, MCMC, Hybrid, Custom).

TECHNICAL

Collapse status per category with different defaults: the curated Capture Class group starts **expanded**, Classic, MCMC, Hybrid, and Custom start **collapsed**. The status is not persisted — on app restart all categories are back to the default state. The chevron arrow rotates animated. The number on the right in the header shows the count of presets in that category. The click hit area spans the entire header region.

IN PLAIN WORDS

Expand and collapse categories to keep the preset list tidy. At app start the Capture Class group is open, Classic, MCMC, Hybrid, and Custom are closed. Click on the header (the entire region is clickable) and the list slides open or shut with a brief chevron animation. The small number on the right shows how many presets are in the category. After app restart, the default state is back — the app deliberately doesn't persist this collapse setting.

Training Configuration section (I12–I22)

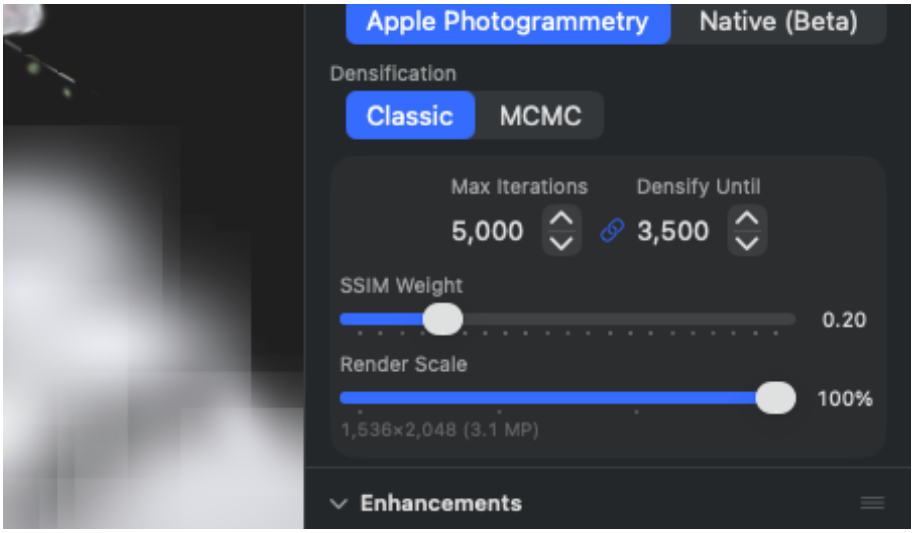

Figure 11: Crop of only the Training Configuration section — Camera Alignment (Apple Photogrammetry active, Native (Beta) inactive), Densification (Classic active), Max Iterations 5,000 / Densify Until 3,500 with link icon, SSIM Weight slider 0.20, Render Scale slider at 100 % (1,536×2,048 = 3.1 MP)

This is where the central levers live: which SfM backend should compute, how densification works, how many iterations, how much SSIM weight. With the MCMC strategy, two additional toggles appear ("MCMC Quality" and "Auto-scale by scene") that are hidden in Classic mode. With the Native SfM backend, the FOV override field is added, which is only needed for video frames without EXIF focal length.

I12 Camera Alignment picker

WHERE

Inspector → Training Configuration → Camera Alignment (segmented picker at the top).

TECHNICAL

Segmented picker with two options: Apple Photogrammetry and Native (Beta). The selection determines the SfM backend used on the next camera reconstruction. At the same time it influences which further Inspector elements are visible: Native additionally shows the FOV override (I13), which is only needed with EXIF-less video frames. Note: for very large outdoor captures you can import the result of an external tool (Metashape or COLMAP) via workspace import — see Chapter 1 (M5) and Chapter 9 (Q3, Q6).

IN PLAIN WORDS

Here you choose how the camera positions are reconstructed — the most important switch for the final quality. Apple Photogrammetry is the fast standard and is enough for most object scans. Native (Beta) is the App-Store-compliant in-house development, good for orbits and turntable scenes, and needs the FOV override (I13) with EXIF-less video frames. With very large outdoor sets you can alternatively compute cameras in Metashape or COLMAP and load the result via workspace import. Details and recommendations per scene type can be found in Chapter 9.

I13 FOV Override field (Native SfM)

WHERE

Inspector → Training Configuration → FOV Override (only visible with Camera Alignment = Native).

TECHNICAL

Numeric text field (range 0-170°), default 0 = automatic determination from EXIF or heuristic. Manual input is needed when the input images were extracted from a video that contains no focal length metadata. Typical values: iPhone Wide ≈ 73°, DJI Mavic Wide-Crop ≈ 70°, drone with full-frame sensor ≈ 84°. The value is clamped to [0, 170] — values outside are pushed back directly. Only affects the native SfM pipeline (Q4/Q5); Apple Photogrammetry ignores this value completely.

IN PLAIN WORDS

If your images have no EXIF (typical with extracted video frames), enter the camera's horizontal field of view in degrees here. Rule-of-thumb values: iPhone Wide ≈ 73°, DJI Mavic Wide- Crop ≈ 70°, drone with full-frame sensor ≈ 84°. A 0 lets the app guess itself — that often goes well, but can go wrong with rare lenses. Values above 170° are automatically clamped back. The field is only visible and only effective if you've chosen Native as Camera Alignment (I12) — Apple Photogrammetry ignores it completely.

I15 Densification picker

WHERE

Inspector → Training Configuration → Densification (segmented picker, always visible).

TECHNICAL

Switches between the two densification strategies: Classic (original 3DGS procedure with clone/split/prune and gradient threshold) and MCMC (Stochastic Gradient Langevin Dynamics with Relocation, NeurIPS 2024). When switching from Classic to MCMC, the app sets the MCMC-specific fields automatically to proven default values (reg weights = 0, MCMC cap multiplier 3.0, sample/noise schedule). Without this automatic initialization, sessions with old presets suffered from the 1.4.4 MCMC collapse bug (460K→5 Gaussians, watchdog kill). The picker selection additionally determines which Inspector elements are visible — with MCMC, I16/I17 appear. Detailed field effect in Chapter 6, T11–T16 (Classic) and T61–T73 (MCMC).

IN PLAIN WORDS

The central strategy choice for growing the Gaussian count. Classic is well tuned from 459 experiments, produces fast and high-quality results, and doesn't require knowing any MCMC fields. MCMC is the newer approach (NeurIPS 2024), more reproducible and foregoes manual threshold adjustment — but it computes about 6× longer for comparable quality. When switching to MCMC, the app sets safe defaults automatically so that the training doesn't run into the 1.4.4 collapse. Details on the strategy fields are in Chapter 6 (T11–T16 Classic, T61–T73 MCMC).

I16 MCMC Quality toggle

WHERE

Inspector → Training Configuration → MCMC Quality (only with Densification = MCMC).

TECHNICAL

Switches gradient accumulation to 2 steps (active) or 1 step (inactive). Accumulates the gradients from two consecutive camera views before the optimizer step is executed. Empirically (Session 33, V544a) reduces the final L1 error by about 6% (0.0246 with Quality vs 0.0261 without, in 3-trial average on Horse-Full-MCMC). The price: doubled training time. With very long trainings (200K iterations) this leads to an additional 10+ minutes waiting time — so only worth it if the last few percent of quality really are needed. Only affects training, not the export format or the viewport display.

IN PLAIN WORDS

Quality mode for MCMC with gradient accumulation across two views. Makes the final result empirically about 6% better (L1 0.0246 instead of 0.0261 in the Horse test), but costs twice as long. If you're already running a 200K MCMC training (easily 2 hours), nearly another hour gets added. Worth it for final showcase renderings or at the end of a quality-sweep session, less so in the daily workflow. Only visible when Densification is set to MCMC (I15).

I17 Auto-scale by scene toggle

WHERE

Inspector → Training Configuration → Auto-scale by scene (only with MCMC).

TECHNICAL

When active, scales the effective max-Gaussians ceiling with the SfM init point count × MCMC cap multiplier (default 3.0). Example: SfM yields 250K init points, base cap = 150K, multiplier 3.0 → effective ceiling = max(150K, 750K) = 750K. When deactivated, only the base applies strictly. Was introduced for v1.4.5 because large outdoor captures with over 1000 frames and correspondingly high SfM point density starved densification with the rigid 150K cap default — superfluous points remained, new ones were not allowed to emerge. Default OFF in custom presets, ON in MCMC built-ins. Only affects training time, not export.

IN PLAIN WORDS

Lets the maximum number of Gaussians grow with the scene size (more precisely: with the number of SfM init points). With small scenes you barely notice a difference, with large outdoor scenes it's often crucial for quality — otherwise the training "suffocates" because the default ceiling of 150K is way too low for the scene. Was introduced for v1.4.5 after very large outdoor sets (over 1000 frames) visibly hit the cap. With the MCMC built-in presets it's switched on in advance; in your own presets it's off by default.

I18 Max Iterations stepper

WHERE

Inspector → Training Configuration → GroupBox → Max Iterations.

TECHNICAL

Stepper with range 1,000–100,000, step size 1,000. Determines the total number of optimizer iterations. Linearly correlated with training time (halving = approx. 50% time). Empirical sweet spots: 20K (Classic Balanced, L1≈0.028), 40K (Classic Quality, L1≈0.023), 200K (MCMC Full, L1≈0.0246). Above 40K with Classic brings on average barely any improvement — diminishing returns. When changing, if the link function (I19) is active, Densify Until is proportionally pulled along (default ratio: 0.5, i.e. Densify-Until = Max/2).

IN PLAIN WORDS

How many training steps are run — more is better, but also costs linearly more time. Rule of thumb: 20,000 for good quality, 40,000 for the optimum with the Classic strategy (above that on average it brings barely anything more). MCMC needs significantly more, 200,000 is the standard here. Doubling the iterations roughly doubles the training time. With the link button (I19) active, Densify Until is proportionally pulled along — practically always what you want.

I19 Link/Unlink button (Densify ↔ Iterations)

WHERE

Inspector → Training Configuration → GroupBox → small link button between Max Iterations and Densify Until.

TECHNICAL

Toggle button that freezes the ratio of Densify Until to Max Iterations. When active (link icon highlighted), on every change of Max Iterations the Densify Until is proportionally pulled along. When unlinked (link-plus icon), the values remain independent. Default is linked, because that reflects the typical correlation — when you pull the training to double the iterations, you usually want to also let densification run proportionally longer. The ratio is computed from the current value when setting the link button; a typical ratio is 0.5 (Densify-Until = half the iteration count).

IN PLAIN WORDS

Small clip button between Max Iterations and Densify Until. When active (link icon highlighted), the two values move together — if you double the iterations, Densify Until also doubles in the same ratio. When not (`link.badge.plus` icon), you can set them independently. Default is linked, because that reflects the typical correlation — longer training usually wants a longer densification phase. For 99% of cases leave it locked.

I20 Densify Until stepper

WHERE

Inspector → Training Configuration → GroupBox → Densify Until.

TECHNICAL

Stepper with range 500–50,000, step size 500. Determines the iteration index from which no new Gaussians are added via clone/split (Classic) or relocation (MCMC) anymore. After reaching it, only position and color are refined. Higher values = more Gaussians = larger file, longer per-iteration time (+30-60% GPU time per step). Typical values: 15K (for 30K max-iter), 20K (for 40K), 100K (for 200K MCMC). With link active (I19), automatically scaled along. Acts differently with Classic vs MCMC: Classic completely stops growth, MCMC stops the relocation logic, but sample/noise adaption continues to run.

IN PLAIN WORDS

Up to which iteration new Gaussians may be added — with Classic via clone/split, with MCMC via relocation. After that it's only about color and shape refinement of the existing points. Higher = more detail, but also larger file and +30-60% GPU time per step. Typical values: 15K (for 30K max-iter), 20K (for 40K), 100K (for 200K MCMC). Normally hangs via link (I19) to Max Iterations — rarely makes sense to decouple that manually.

I21 SSIM Weight slider

WHERE

Inspector → Training Configuration → GroupBox → SSIM Weight.

TECHNICAL

Slider 0.0–1.0 in 0.05 steps, displayed as "0.20". Mixes L1 loss (0.0) and SSIM loss (1.0). L1 tightens brightness per pixel, SSIM tightens structural similarity (edges, local statistics). Default 0.2 is the value from the original 3DGS paper (Kerbl 2023) and reverse-engineered as a robust compromise in numerous sessions. Higher values (0.5+) favor detail preservation, but may ignore local brightness errors. Lower values (< 0.1) lead to detail loss at sharp edges. The SSIM computation runs in the shader with an 11×11 Gaussian window. Performance: at 0.0 (only L1) the training is approx. 8-12% faster, because the SSIM computation is skipped in the shader.

IN PLAIN WORDS

How strongly the structural image similarity (edges, local patterns) is weighted against the pure brightness comparison. 0.2 is the standard from the original 3DGS paper and is enough for almost all scenes. Higher (0.5+) for fine structures like hair, fur, or vegetation — there more structural weight helps. Lower (0.0) makes training about 8-12% faster, because the SSIM computation in the shader is skipped, but costs detail at sharp edges. If you don't have a good reason for a change, leave 0.2 as is.

I22 Render Scale slider

WHERE

Inspector → Training Configuration → GroupBox → Render Scale.

TECHNICAL

Slider 0.25–1.0 in 0.25 steps, displayed as "100%". Scales the training rendering resolution relative to source image size. Biggest lever on performance: 50% reduces GPU time by approx. 75% (because 4× fewer pixels), 25% by approx. 94%. The gradient threshold is automatically scaled along. Below the slider a live resolution display in MP appears (e.g. "2304×1296 (3.0 MP)"). If the current value deviates from the recommended, "— recommended: 50%" is shown in orange text. The recommendation targets ~3 MP effective resolution — the range most efficiently processed by Apple Silicon GPUs. 4K source images get e.g. 25% recommended automatically, FullHD images 100%. A change additionally triggers the buffer reallocation.

IN PLAIN WORDS

What resolution training renders at — one of the biggest performance levers. Full (100%) gives the best quality, but costs a lot of GPU time on large images. Half (50%) saves about 75% GPU time, because four times fewer pixels are computed — perfect for 4K sources. Below the slider you see the effective resolution in megapixels; the app targets about 3 MP, because that runs most efficiently on Apple Silicon. If your value deviates from that, the app shows an orange "recommended" hint — usually it's worth following.

Enhancements section (I26–I29, I42–I44)

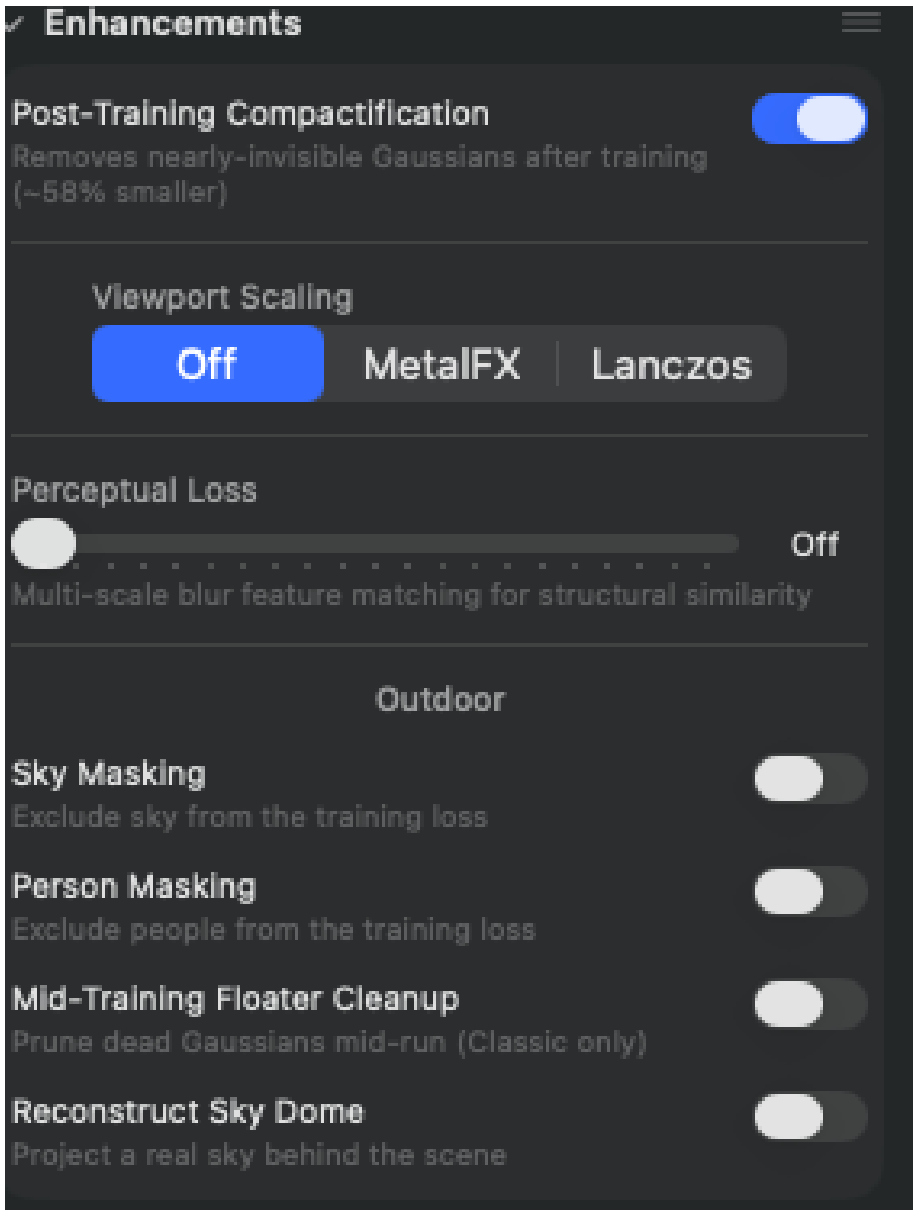

Figure 12: Crop of only the Enhancements section — three rows: Post-Training Compactification (toggle on), Viewport Scaling (segmented picker Off/MetalFX/Lanczos), Perceptual Loss (slider on "Off"). Each row with subtitle explains its function

The Enhancements section groups three features that improve image quality without changing the core training loop itself. The first two (I26-I27) are **post-training** or **viewport stages**: Compactification tidies up after training ends, the Viewport Scaling is a pure viewport renderer that doesn't influence the running training. The Perceptual Loss (I29) despite belonging to the section is a training component — it is activated as an additional loss term during training, hence the separation from the viewport controls via a divider. As of v1.6 the section also has an **Outdoor** group (I42–I44: Sky Masking, Mid-Training Floater Cleanup, Reconstruct Sky Dome) — training options against sky floaters that used to live in the Settings window and now sit here per project.

I26 Post-Training Compactification toggle

WHERE

Inspector → Enhancements → Post-Training Compactification.

TECHNICAL

Activates the V443 post-processing: after completion of training iterations, Gaussians with opacity below 0.01 (1% visibility) are deleted. Empirically this reduces file size by ~55-58% with zero visible quality loss — because these Gaussians don't visually contribute anyway. The compactification runs as a GPU compact pass and takes fractions of a second to a few seconds depending on Gaussian count. Doesn't affect training performance. If this toggle is off, invisible Gaussians are also exported — only relevant if you want to use the format for another training stage (Continue Training), otherwise wasted storage.

IN PLAIN WORDS

Cleans up Gaussians after training that you can't see anyway (opacity below 1%). Makes the export files about half the size (55-58% size reduction) without visible quality loss. Runs as a brief GPU pass after the last iteration, takes only fractions of a second to a few seconds. Should practically always be on — the only reason to switch this off is if you want to continue training later via Continue Training and need to keep invisible Gaussians too. With normal export workflows just leave it on.

I27 Viewport Scaling picker

WHERE

Inspector → Enhancements → Viewport Scaling (segmented picker with three options: Off, MetalFX, Lanczos).

TECHNICAL

A single segmented picker that selects the viewport upscaler — the three options are **mutually exclusive**. When the training resolution (via I22 Render Scale) is lower than the viewport size, the chosen mode scales the rendered frame up to display size. **Off** = plain bilinear stretch. **MetalFX** = Apple's ML-based Spatial Upscaler, the sharpest option (the ML model is optimized for sharp edges), overhead approx. 1-2ms per frame on M3 GPUs. **Lanczos** = Apple's Metal Performance Shaders with 8-tap sinc resampling, classical without ML, minimal overhead (< 0.5ms), quality below MetalFX, but without the ML-typical "smoothing" of fine line structures. The renderer pipeline is reconfigured live when switching — visible immediately, without restart. **Background:** formerly these were two separate toggles (MetalFX + Lanczos) that could be on at the same time — a contradictory state in which MetalFX silently overrode Lanczos. The picker removes that state; a "both-on" state possibly inherited from older sessions self-heals to MetalFX on the next switch. Acts **only** on the live viewport, not on rendered exports (orbit video, screenshots) — those are rendered at full source resolution.

IN PLAIN WORDS

Sharpens the live image in the viewport — especially useful if you're working with reduced training resolution (Render Scale 50%, see I22). Three steps, of which only one is ever active: "Off" simply stretches the pixels, "MetalFX" uses Apple's machine learning for the sharpest edges (practically always the best choice), "Lanczos" is the classical filter without ML — take it as a fallback in case MetalFX smooths lines or shows artifacts in a scene. Takes effect live, without restart. Only acts in the live viewport, not on exported orbit videos or screenshots — those are rendered at full source resolution. Unlike before, you can no longer accidentally pick two modes at the same time.

I29 Perceptual Loss slider

WHERE

Inspector → Enhancements → Perceptual Loss.

TECHNICAL

Slider 0.0–0.2 in 0.01 steps, displayed at 0.0 as "Off", otherwise as "0.05" etc. Activates an additional loss term that compares multi-scaled Gaussian blur of the rendering with the ground truth image (3 blur scales). Captures structural differences that L1+SSIM alone don't detect. V460 implementation. Empirically, a value of 0.05-0.1 improves the L1 score in sessions by a few percent, but costs ~5% training time (additional forward pass through the blur kernels). Above 0.15 the training becomes unstable and L1 worsens again (loss term dominates the optimization). Acts **during** training, not in post-processing — despite its position in the "Enhancements" section it is not a pure post-finishing.

IN PLAIN WORDS

An additional loss term that checks structural image similarity over three different blur stages. Especially helps with scenes with fine structures like hair, fabric, or vegetation, because it captures patterns that L1+SSIM alone don't see. Smaller values are safer — 0.05 to 0.1 is the sweet spot, above 0.15 training becomes unstable and the loss worsens again. At 0 (Off) the function is completely off and costs nothing; active it eats about 5% training time for the extra forward pass through the blur kernels. Acts despite the "Enhancements" section directly during training, not just in post-processing.

I42 Sky Masking

WHERE

Inspector → Enhancements (Outdoor group) → Toggle “Sky Masking”. Bound: `AppState.trainingConfig.skyMaskingEnabled` (per project, `@DefaultFalse`). Default: `false`.

TECHNICAL

Activates pre-training Apple Vision-based sky pixel segmentation. Before training starts, the sky region is extracted for each input camera via Apple Vision foreground mask (Sky = Background) and assigned as a per-pixel mask to the respective camera. During training, the loss contribution per pixel is multiplied by the complement of the Sky Masking — sky pixels contribute 0 to the gradient, so Gaussians projecting into the sky receive no optimization signals and therefore do not become “denser” or “brighter”. Significantly reduces floaters (dark clumps in the sky) in outdoor/drone scenes. Costs ~3% L1 regression in classic 40K training (see `memory/dev_outdoor-floater-reduction.md`). Only useful for outdoor scenes with clearly recognizable sky; for indoor scenes or white backgrounds the sky segmentation identifies wrong areas and blocks valid loss signals. The value is now stored per project (no longer app-global) and follows the preset / scene file.

IN PLAIN WORDS

In outdoor shots with sky in the frame, black or colored clumps often form in the sky — so-called “floaters”. This option automatically detects where the sky is and tells the training: “Leave the sky alone.” Works very well for drone flights and landscape scenes. For indoor or dark backgrounds it can make the image worse — so only enable it when real sky is visible. Details: `memory/dev_outdoor-floater-reduction.md`.

I43 Mid-Training Floater Cleanup

WHERE

Inspector → Enhancements (Outdoor group) → Toggle “Mid-Training Floater Cleanup”. Bound: `AppState.trainingConfig.floaterCleanupEnabled` (per project, `@DefaultFalse`). Default: `false`.

TECHNICAL

Enables two additional density-control passes during Classic 40K training (preset “P4 Quality”): at iteration 20,000 and at iteration 30,000. Both passes scan all Gaussians by three criteria: (a) very low opacity (default 0.005), (b) tiny screen-space size, (c) no loss contributions in the last 1000 iterations. Gaussians that meet all three conditions are purged. Effect: ~5–15% fewer Gaussians at the end of training, visibly fewer dark clumps in the sky for drone/outdoor scenes. Costs ~1–3% L1 regression on close-up indoor scenes, hence not enabled as default. The two cleanup iterations (20K, 30K) are hard-coded and currently cannot be changed via UI; for shorter trainings (e.g. P2 Preview 5K) the toggle has no effect, because it never reaches the iteration marks. **New:** the toggle is only operable when the active preset uses the **Classic** densifier (`densificationStrategy == .classic`). Under MCMC or Hybrid it is **disabled** and an inline hint appears, because those strategies handle dead gaussians themselves anyway (MCMC via relocation, Hybrid via combined reloc/noise logic) — the manual cleanup passes would be ineffective or counterproductive there. Code reference: `RadianceKitApp.swift`, General tab. Details: `memory/dev_outdoor-floater-reduction.md`.

IN PLAIN WORDS

During training, “dead” Gaussian points sometimes form that no longer contribute to image quality but occupy memory. This option cleans up twice during a long training (at 20K and 30K iterations) and removes these corpses. For outdoor scenes with sky this is particularly useful, because most floaters collect there. For small trainings or close-ups of furniture, not really necessary. The switch can only be turned on when your preset uses the Classic densifier — with MCMC or Hybrid presets it is grayed out (with a short explanation), because those clean up their dead points themselves.

I44 Reconstruct Sky Dome

WHERE

Inspector → Enhancements (Outdoor group) → Toggle “Reconstruct Sky Dome”. Bound: `AppState.trainingConfig.skyDomeEnabled` (per project, `@DefaultFalse`). Default: `false`.

TECHNICAL

Activates the pre-training Sky Dome projection (V549e MVP). After SfM and before training start, the Apple Vision sky mask shared with S7 is extracted from the image for each input camera, the sky pixels are un-projected using the camera intrinsics onto a virtual sphere surface (default radius 8× scene radius). On this sphere ~5000 new Gaussians are initialized with color means from the projected sky pixels, very large scale (1.0 in scene units) and initial opacity 0.95. These 5000 Gaussians are not a sky mask in the classical sense — they are trained like all others, but the high initial opacity keeps them in a thin shell. Result: for 360° novel views in outdoor/drone scenes, actual sky color and cloud structures appear instead of dark confetti clumps. The value is remembered across restarts. Only useful for outdoor scenes with at least 360° camera coverage; for pure object captures without sky view it has no effect. Status: experimental, broader A/B validation across more outdoor sets still pending.

IN PLAIN WORDS

Instead of having training try to “guess” the sky from the few visible pixels (which leads to floaters), RadianceKit projects the sky pixels directly onto a virtual sphere around the scene before training starts. When you then rotate the finished scene in 360°, you see real sky instead of black clumps. Only works for outdoor shots in which sky is actually in the frame. For living room scans or studio shots it brings nothing.

Metrics section (I30–I38)

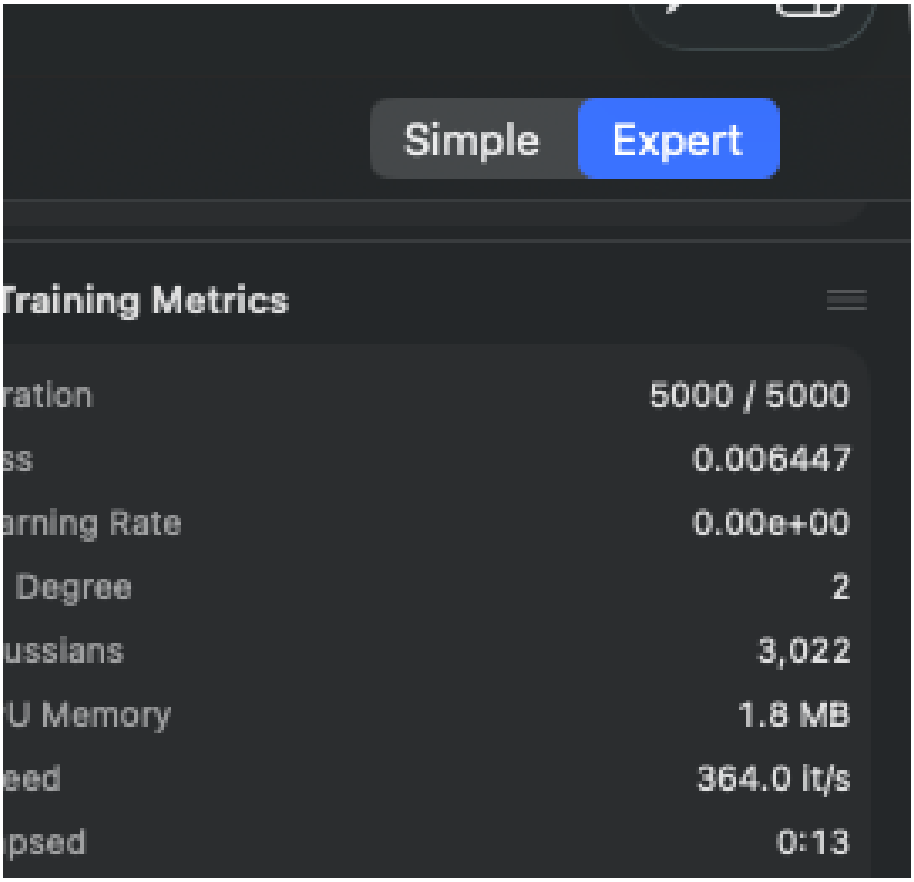

Figure 13: Crop of only the Training Metrics section after completed training on Bouquet (5K iterations, 2,991 Gaussians final) — table with training metrics (Iteration, Loss, SSIM Loss, Combined Loss, Gaussian Count, Learning Rate, Elapsed, ETA)

While a training is running, the Metrics section shows nine live values from the training loop. Before a training starts the section is empty (“Start training to see live metrics”). All values are updated every ~30 iterations (update frequency of the stream). The section is read-only — no element is clickable or modifiable. For deeper analysis, consult the JSONL training logs under `~/Documents/RadianceKit/Logs/` (script `python3 scripts/analyze_logs.py best 5`).

I30 Iteration

WHERE

Inspector → Metrics → Iteration. Read-only.

TECHNICAL

Displayed in the format “4523 / 40000” — current iteration over total planned iterations. Counts synchronously with the training loop, which pushes the values every ~30 iter. The second number corresponds to the Max Iterations value at the start time; it doesn’t change anymore, even if the user adjusts the stepper afterwards — the running run uses its own snapshot copy. If the app pushes additional iterations via the Training menu (Continue Training +5K/+10K/+20K), the denominator increases.

IN PLAIN WORDS

Where training currently stands. “4523 / 40000” means: 4523 of 40,000 steps are through, so about 11% done. The left number counts up in real time; if it stays stuck for minutes, training is hung — usually a hint of GPU throttling or a competing app. The right number corresponds to the Max Iterations value (I18) at training start and doesn’t change anymore, even if you adjust the stepper later. With Continue Training (+5K/+10K/+20K) it grows by the additional steps.

I31 Loss

WHERE

Inspector → Metrics → Loss. Read-only.

TECHNICAL

Float value with six decimal places (e.g. “0.024385”). Measures the combined L1+SSIM loss (mix controlled via I21 SSIM Weight) plus optional Perceptual Loss (I29) and other regularizers. Scale is not absolute, but scene-dependent — demands the same dataset for most comparisons. Typical end values with good configurations: - Classic Quality 40K iters: 0.022–0.025 (Horse, Truck, Garden) - MCMC Full 200K iters: 0.024–0.028 - Outdoor drone 30K: 0.030–0.060 (geometry-related worse) - Indoor apartments: 0.018–0.025

Values above 0.10 after 5K iterations indicate SfM problems (bad camera poses) — abort and recompute SfM.

IN PLAIN WORDS

How far the rendered image still deviates from the original — combined from L1, SSIM, and possibly Perceptual Loss. Smaller is better. Below 0.03 is usually really good, below 0.05 still okay, outdoor scenes lie geometry-related rather at 0.03-0.06. Above 0.10 after several thousand iterations is a warning signal — mostly it’s the camera reconstruction (SfM didn’t work cleanly). The scale is not absolute, but scene-dependent; only make comparisons within the same dataset. If the number suddenly jumps up, usually a gradient explosion event occurred.

I32 Learning Rate

WHERE

Inspector → Metrics → Learning Rate. Read-only.

TECHNICAL

Scientific notation display (e.g. “1.60e-04”). Current learning rate for the position parameters (3DGS has six independent LRs for position, SH-DC, SH-Rest, opacity, scale, rotation — displayed here is the position LR as a representative value). Default starting value 1.6e-4, which decays via an exponential decay to ~1.6e-6 by training end. The decay is adjustable via the LR schedule field in the training configuration (T field in Chap. 6). If the LR stays unusually high (e.g. 1e-3 or more after 10K iterations), this could indicate a faulty loaded configuration.

IN PLAIN WORDS

How big the optimization steps currently are — specifically the learning rate for the Gaussian positions. Starts at 1.60e-04 and sinks exponentially to about 1.60e-06 by training end (“1.60e-06” = 0.0000016). The curve runs automatically, you don’t have to adjust anything here. If the value after 10,000+ iterations is still greater than 1e-3, probably a faulty config was loaded — abort training and pick a new preset. Internally 3DGS has six independent learning rates (position, SH-DC, SH-Rest, opacity, scale, rotation); here you only see the position LR as a stand-in.

I33 SH Degree

WHERE

Inspector → Metrics → SH Degree. Read-only.

TECHNICAL

Integer 0-3. Spherical-harmonics degree for the color representation. Starts at 0 (only the DC component, i.e. direction-independent color per Gaussian — so only a single RGB constant) and rises progressively to 3 over the course of training. The standard schedule lifts the degree at 1000/2000/3000 iterations by 1 each. SH-3 corresponds to 48 color coefficients per Gaussian (3 RGB channels × 16 SH basis functions). Higher SH degree = more direction-dependent reflection (glossy surfaces look correctly different under different viewing angles), but also more memory and slower training.

IN PLAIN WORDS

How complex the color representation per Gaussian currently is. Starts at 0 (only a direction-independent color per point) and is stepwise raised to 3 — typical at iteration 1000, 2000, and 3000. Stage 3 means 48 color coefficients per Gaussian and allows direction-dependent reflections, i.e. that glossy surfaces look correctly different from various viewing angles. You don’t need to touch this actively, the schedule runs automatically. Higher degree costs more memory and slows training slightly — but that’s the price for realistic highlights.

I34 Gaussians

WHERE

Inspector → Metrics → Gaussians. Read-only.

TECHNICAL

Current number of Gaussians in the model, formatted with locale separator (e.g. “524.318”). Growth: - Classic: starts at the SfM init points (typically 50K-300K), grows via clone/split until just before Densify Until, then static until training end (modulo pruning) - MCMC: sample points are added up to the MCMC cap, then only relocation

Healthy end values: - Classic Quality: 400K-700K (Horse 524K, Garden 800K) - MCMC Full: exactly on the cap (default 150K, with auto-scale multiplier × SfM count, depending on scene 500K-1.5M)

With MCMC, if the number falls to < 60% of the cap → anomaly (collapse indicator, indicates too aggressive regularizers).

IN PLAIN WORDS

How many Gaussian points the 3D model currently has. Grows during training until Densify Until (I20) is reached; after that the number stays practically constant. More points = more detail, but also larger file and slower rendering in the viewport. 500,000 Gaussians is a typical mean for Classic Quality on a medium scene; MCMC Full lands depending on auto-scale (I17) between 500K and 1.5M. If the number with MCMC suddenly falls below 60% of the cap, it’s a collapse indicator — usually too aggressive regularizers.

I35 GPU Memory

WHERE

Inspector → Metrics → GPU Memory. Read-only.

TECHNICAL

Estimate of the Gaussian buffer memory consumption as Gaussian count × 616 bytes (formatted in memory style). 616 bytes is the empirical size of a fully equipped Gaussian (position, scale, rotation, opacity, SH coefficients degree 3, gradient accumulator). The display does **not** capture the renderer overhead (tile buffer, sort buffer, backward buffer) — the real GPU memory requirement is typically 2-3× over this value. At 500K Gaussians: displayed ~290 MB, real ~700 MB. At 1.5M Gaussians: displayed ~880 MB, real ~2.5 GB. On M3 Max with 64+ GB Unified Memory uncritical, on M3 Pro with 18 GB already a limit.

IN PLAIN WORDS

An estimate of how much GPU memory the Gaussians themselves occupy — around 616 bytes per point. The actual GPU consumption is 2-3× higher than displayed, because the renderer adds its own tile, sort, and backward buffers. On a MacBook with 16-18 GB unified memory you should stay below 500K Gaussians; with M3 Max or Studio (64+ GB) you can easily run 1.5M and more. If training suddenly crashes or the system swaps, usually the limit is reached here — Render Scale (I22) down or Densify Until (I20) reduce.

I36 Speed

WHERE

Inspector → Metrics → Speed. Read-only.

TECHNICAL

Iterations per second with one decimal place (“24.3 it/s”). Computed by the trainer as a moving average over the last ~100 iterations. Typical values: - Quick preset (1K iters): 80-120 it/s (short, no steady state) - Classic 20K @ 1.0 Render Scale (Truck scene, M3 Max): 25-35 it/s - Classic 20K @ 0.5 Render Scale: 80-120 it/s - MCMC 200K @ 0.5 Render Scale: 25-50 it/s (slower due to relocation) - With 1M+ Gaussians and full resolution: < 10 it/s

Decreasing speed over the course of training is normal — more Gaussians = more compute per iteration. Sudden drops (e.g. from 30 → 5 it/s) indicate GPU thermal throttling or competing apps.

IN PLAIN WORDS

How fast training is running, in iterations per second. Typically stands at 20-50 it/s, with reduced Render Scale (50%) and small scenes also 80-120 it/s. Sinks over the course of training quite naturally, because more Gaussians = more work per iteration. Sudden drops (e.g. 30 → 5 it/s) indicate GPU thermal throttling or competing apps — browser tabs with video, Time Machine backup, Photos indexing. Keep the app in the foreground and close background programs often helps. With 1M+ Gaussians and full resolution, below 10 it/s is normal.

I37 Elapsed

WHERE

Inspector → Metrics → Elapsed. Read-only.

TECHNICAL

Already elapsed time as “4:23” (m:ss) or “1:23:45” (h:mm:ss). Format switch from 1 hour. Measures only the pure training time, not the upstream phases (SfM computation, image import). With pause/resume the clock keeps running — so it’s wall-clock, not CPU time.

IN PLAIN WORDS

How long training has been running, as a pure stopwatch (wall-clock time). Format is “m:ss” until one hour, after that “h:mm:ss”. Not “CPU time”, but “how long have we been waiting” — so pause times also count. Measures only the pure training phase, not the upstream SfM computation or image import. Useful for comparison with the ETA (I38) — if Elapsed clearly overshoots the original ETA, training has somewhere become slower than planned.

I38 ETA

WHERE

Inspector → Metrics → ETA. Read-only.

TECHNICAL

Estimated remaining time as “17:42” or “1:12:35”. Computation: (Max Iterations – current iteration) / iterations per second. Shows “–” when speed is currently zero (at the very start or during pause). The estimate is **not** adjusted to the typical slowdown towards training end — especially with MCMC and Classic with large Densify-Until values, the training tends to become slower because more and more Gaussians come into the picture. Real it typically stays 10-20% above the initial ETA.

IN PLAIN WORDS

How much time is expected to remain — computed from the remaining iterations and the current speed (I36). A rough estimate: if the Mac suddenly becomes slower (more Gaussians from the densify phase, thermal throttling, other apps), it can take longer than displayed. The app doesn’t factor in the typical slowdown towards training end, so the real end usually lands 10-20% above the initial ETA. Plus 15% factored in, then it usually fits. Shows “–” when speed is currently 0 (training start or pause).

Loss Chart section (I39–I41)

Figure 14: Crop of only the Loss History section after completed training — Current 0.0064, Min 0.0035 (green), blue curve from 0.027 (iteration 1) to 0.0035 (iteration 5K) with characteristic kink around iter 200, below it Gaussian Count chart orange

The Loss Chart section visualizes the training course over time. It consists of two charts: a Loss Curve chart (large, top, blue) and a Gaussian Count chart (smaller, bottom, orange). Both are built up live during training and persist until the next training start. Before the first training the area is empty (“Loss curve will appear during training”). The charts are pure SwiftUI path drawings (not Swift Charts framework), so they render fluidly even with 100K+ points.

I39 Current Loss (display)

WHERE

Inspector → Loss Chart → left label area “Current: 0.0287”. Read-only.

TECHNICAL

Float value of the last loss sample point, formatted with four decimal places. Identical with I31 (Loss in the Metrics section), just here more compactly formatted. Source is the loss history — a list that gets an entry every ~30 iterations. Only finite values are added to the list — NaN/Infinity (very rare, in the case of a gradient explosion bug) are filtered.

IN PLAIN WORDS

The current loss value in shorter notation than in the Metrics section (four decimal places). Content-wise identical with I31, but here the display sits directly at the loss chart and gives you the exact numeric value when observing the curve. Like all live metrics, it’s updated every 30 iterations. NaN or Infinity values (extremely rare with gradient explosion bugs) the app automatically filters out. Useful when looking at the chart so you don’t have to jump to the other section.

I40 Min Loss (display)

WHERE

Inspector → Loss Chart → right label area “Min: 0.0245” (green). Read-only.

TECHNICAL

Minimum of all loss values ever seen of the current training run. Live recomputed from the loss history — no separate persistence. Displayed with green text, because “Min” = “Best so far”. The dashed green line at the lower chart edge marks this Y position visually. With continue-training sessions, the min tracking restarts — the old history is replaced in the UI by the new (not appended). If the current training runs worse than the previous, the min display can be larger than the previous end result.

IN PLAIN WORDS

The lowest loss value this training has seen so far — shown in green, because “best so far”. The dashed green line at the lower chart edge marks this position visually too. If the current curve currently lies clearly above it, there’s hopefully still an improvement; usually Min is the end-result indicator that matters to you later. With continue-training sessions, the min tracking restarts, because the old history is replaced in the UI by the new — the min value can therefore look worse than the previous end result.

I41 Gaussian Count Chart

WHERE

Inspector → Loss Chart → second chart below it (orange). Read-only.

TECHNICAL

Line chart of Gaussian count over the training iterations. Source: the Gaussian count history (list of (iter, count) pairs, filled by the trainer every ~30 iter). Y scale dynamic between minimum and maximum of the history. With Classic strategy the curve typically looks like this: steadily rising up to Densify Until, then flat (with small pruning fluctuations). With MCMC: steep rise up to cap, then horizontal line (relocation keeps the number constant). If the curve **falls** despite active training, densification prunes too aggressively — indication of wrong defaults or a known MCMC collapse bug (v1.4.4 hotfix topic).

IN PLAIN WORDS

How the number of Gaussians develops over the training time — the smaller orange chart below the Loss curve. With Classic strategy, the line rises steadily until Densify Until (I20) is reached, after that it stays flat with small pruning fluctuations. With MCMC it shoots steeply up to the cap and then stays horizontal, because relocation keeps the number constant. If the curve, despite active training, suddenly kinks downward, densification is too aggressive with pruning — classic sign of the MCMC collapse bug from v1.4.4. Then app update or a switch back to Classic helps.

How to read the loss curve?

The loss chart is the most important diagnostic tool in the Inspector — no other indicator shows as directly whether training is usefully progressing or stuck. The typical healthy shape is a quick drop in the first 1000-3000 iterations (from ~0.15 to ~0.05), followed by a slow, steady decline until training end (to 0.020-0.030). Logarithmically the curve looks like a smooth diagonal.

What does a plateau in the loss mean? If the curve stays flat over several thousand iterations, there are two possible readings: (a) training has “converged” — the loss can no longer significantly drop, because the model is as good as it can be with the given data and settings. That’s desired; that’s “done”. (b) Training “hangs” — the loss could actually still drop, but optimization stagnates (local minimum, learning rate too small, densification off). Distinguishing: if the loss value lies in a typically good range (0.020-0.030 with indoor/object, 0.040-0.060 with outdoor) and the curve has been flat since 5K iterations, it’s converged. If the value is clearly higher than with comparable scenes (e.g. 0.08), it’s hung.

Note: Gaussian plateau ≠ Loss plateau. A plateau in the Gaussian count does **not** mean “training is done”. It only means that densification has stopped adding new points — either because is reached (Classic) or because the MCMC cap is full. Training continues afterwards and only refines the existing points. The actual “done” signal you read from the loss curve and from the Iteration display (I30), not here.

Rule of thumb for aborting: If the loss curve after 5000+ iterations lies above 0.08 and barely sinks anymore, with high probability the SfM reconstruction is skewed. Abort training, look in Chapter 9 to see whether the chosen SfM backend matches the scene,

possibly switch to COLMAP/Native, then restart. Better to invest 10 minutes in better SfM than 2 hours of training with bad camera alignment.

When to reach for the Inspector?

Quick reference: which section + which controls for which typical use case?

Common task	Section	Control IDs
Desaturate finished Splat colors	Look	L1 (Saturation)
Round needle/confetti Splats	Look	L2 (Splat length)
Fill holey cloud / enlarge Splats	Look	L3 (Splat size)
Fade distant "far confetti" on orbits	Look	L4 (Fade far region)
Discard look adjustments	Look	L5 (Reset finishing)
Load prefab setup	Presets	I7 (click row)
Save own setup	Presets	I1 → I2 → I4
Share setup with colleague	Presets	I5 (Export) or I6 (Import)
Change SfM backend (e.g. because Apple-PG too unstable)	Training Configuration	I12 (see Chap. 9)
Process video frames without EXIF focal length	Training Configuration	I13 (FOV Override)
COLMAP performance: GLOMAP instead of Classic	Training Configuration	I14
Switch from Classic to MCMC	Training Configuration	I15
Let training run longer	Training Configuration	I18 (Max Iter) + I20 (Densify Until) — coupled via I19
Halve GPU time	Training Configuration	I22 (Render Scale to 50%)
Training quality +6% (MCMC)	Training Configuration	I16 (MCMC Quality)
Outdoor scene with many SfM points	Training Configuration	I17 (Auto-scale by scene)
Set up / change COLMAP path	Training Configuration	I23 / I24 / I25
Make export files smaller	Enhancements	I26 (always leave on)
Sharper viewport without more training time	Enhancements	I27 (Viewport Scaling → MetalFX)
MetalFX smooths too much → alternative	Enhancements	I27 (Viewport Scaling → Lanczos)
Last bit of detail with fine structures	Enhancements	I29 (Perceptual Loss 0.05-0.1)
Monitor training	Metrics	I30 (progress), I36 (pace), I38 (remaining time)
Assess quality early	Metrics	I31 (Loss < 0.05 after 5K = good)
[illegible] stuck	Metrics [illegible] Loss Chart	[illegible] → SfM redo)

CHAPTER

Chapter 3 — Settings

The Settings window opens via `RadianceKit → Settings…` or the standard keyboard shortcut `⌘,`. It contains two tabs: **General** and **AI Helpers**. Unlike the Inspector values from Chapter 2, the settings in this window take effect **app-globally** (across all projects) — they are persisted and survive app restarts. The General tab groups three content sections: Interface, Viewport, and Training. (The three Outdoor-Floater toggles — Sky Masking, Mid-Training Floater Cleanup, Reconstruct Sky Dome — that used to live here moved, as of v1.6, into the Expert Inspector's **Enhancements** section, where they are now stored per project; see Chapter 2, I42–I44.) The AI Helpers tab enables the on-device machine-learning helpers (Vision, CoreML) for SfM and training preprocessing.

Earlier controls for collectively enabling or disabling all AI Helpers no longer exist in the current version — they are therefore not documented here. The earlier "Coming Soon" area for not-yet-shipped helpers has also been removed and is not referenced here.

General Tab

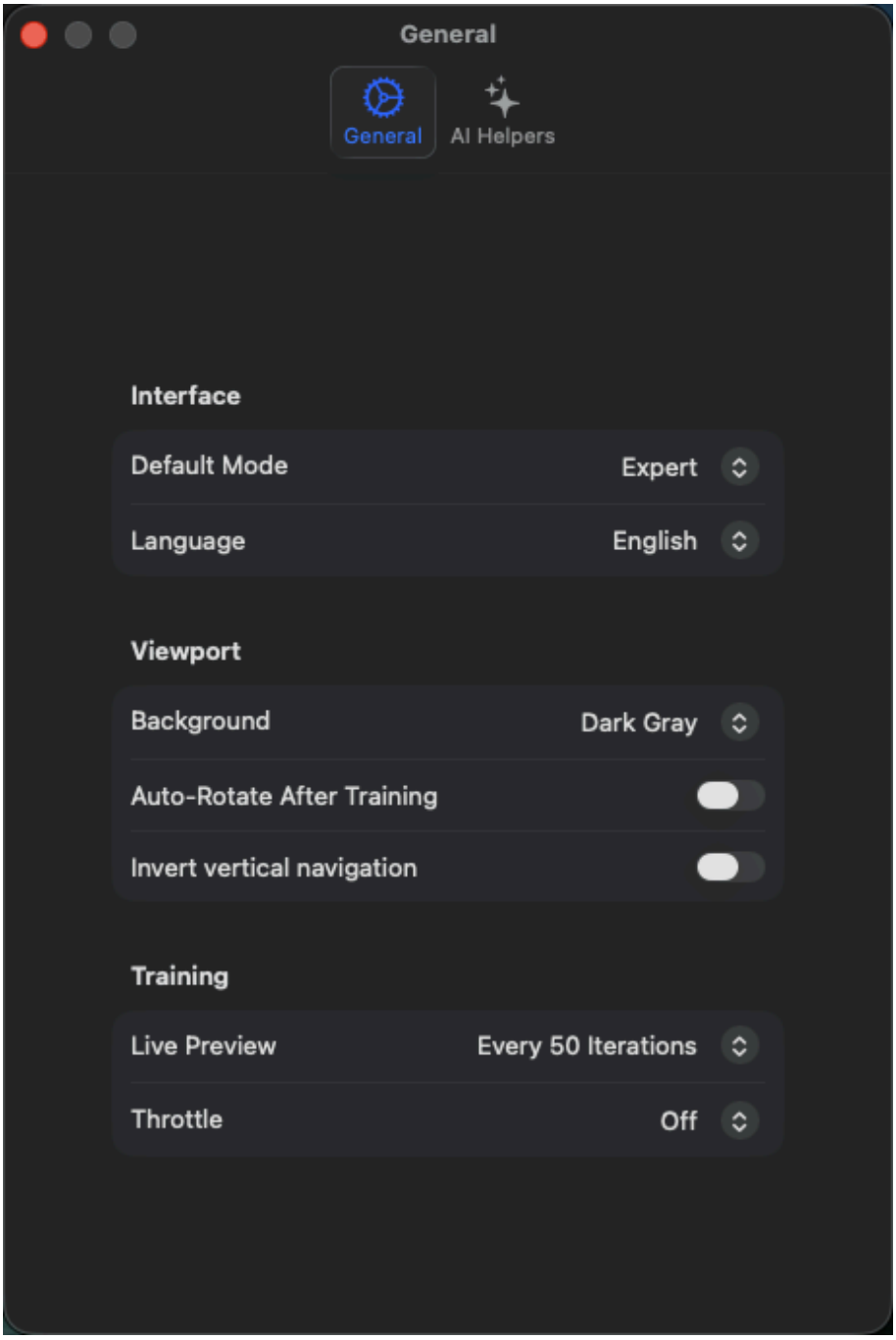

Figure 15: Settings → General tab with Interface, Viewport, Training and Experimental sections

S1 Default Mode

WHERE

Settings → General → Interface → Default Mode picker. Bound:. Default: `.simple`.

TECHNICAL

Controls which of two UI modes the app opens in after the next launch. “Simple Mode” is the guided wizard workflow in 4 steps (Import → Processing → Preview → Export, documented in Chapter 10 under Z1–Z4), “Expert Mode” the classic three-panel layout with Navigator, 3D Viewport and Expert Inspector from Chapter 2. The value is remembered across restarts. Identical effect to the menu `Mode → Simple Mode` (⌘1) / `Mode → Expert Mode` (⌘2), except the menu switches the running session while this picker sets the default for future sessions. Both modes access the same project state — projects, cameras and training configuration are preserved when switching modes. Mode-specific toolbar buttons are re-rendered immediately.

IN PLAIN WORDS

Here you choose which interface RadianceKit boots up with on the next launch. “Simple Mode” is the beginner mode: four clear steps, predefined presets, hardly any options. “Expert Mode” is the full toolbox layout with all controls you see in Chapter 2. You can switch back and forth at any time via the “Mode” menu without losing any images or training progress.

S2 Language

WHERE

Settings → General → Interface → Language picker. Bound:. Default: `.system` (follows the macOS language).

TECHNICAL

Selects the display language of the entire app UI, independent of the macOS system language. RadianceKit is localized in 17 languages (`de` , `en` , `pl` , `en-AU` , `ar-SA` , plus 12 more). When set to “System”, the app follows the macOS language. With an explicit choice the language setting is remembered across restarts; full effect usually requires an app restart, because localization bundles are only loaded at startup. The 298 documented localization keys in the project are all considered, including all texts in sub-views and help tooltips.

IN PLAIN WORDS

If your Mac runs in English but you would rather have the German RadianceKit interface (or vice versa), you set that here. Most texts swap immediately. Some dialogs only appear in the new language after an app restart.

S3 Viewport Background

WHERE

Settings → General → Viewport → Background picker. Bound:. Default: `.darkGray` (RGB 0.1, 0.1, 0.1).

TECHNICAL

Sets the default background color for the 3D Viewport. Three options: “Dark Gray” (RGB 0.1, 0.1, 0.1 — default), “Black” (0, 0, 0) and “White” (1, 1, 1). The setting persists the default for new projects and sessions across restarts and at the same time updates the running Metal renderer immediately. Identical options can be found in the menu `Viewport → Background` (M21, M22, M23), but the Settings picker sets the default, while the menu switches the running display. Important for screenshots and demo videos: white backgrounds emphasize green/blue floaters more strongly, dark backgrounds are better for clean render captures.

IN PLAIN WORDS

The color behind your 3D models in the preview window. Dark gray is the default and fits most scenes. White is good for screenshots, black looks more elegant for render captures. You can switch the color at any time via the menu “Viewport → Background” for the running scene — this setting only defines which color is active again the next time you open it.

S4 Auto-Rotate After Training

WHERE

Settings → General → Viewport → Toggle “Auto-Rotate After Training”. Bound:. Default: `false`.

TECHNICAL

Starts a continuous turntable rotation of the Viewport camera around the scene centroid immediately after training ends (default rotation rate ~0.3 rad/s). Practically useful for demo sessions, A/B comparisons and to immediately assess from a 360° view whether “floaters” have formed at the scene edge. The effect is visually identical to the menu `Viewport → Toggle Auto-Rotation` (M16, ⌘⌥T), except this toggle triggers the behavior automatically after training ends rather than manually. It can be interrupted at any time via the menu or by clicking in the Viewport (which pauses the rotation). Has no influence on training performance — the rotation only runs once training is finished.

IN PLAIN WORDS

When enabled, the 3D scene rotates automatically as soon as training is done — like a carousel. Nice when you do overnight training and want to see the result already in motion in the morning without having to click yourself. For long sessions where you only monitor the training, leave it off.

S5 Live Preview Interval

WHERE

Settings → General → Training → Live Preview picker. Bound: `AppState.trainingConfig.livePreviewInterval`. Default: `0` (Off).

TECHNICAL

Determines the iteration interval at which the running training snapshot is rendered into the 3D Viewport. Four discrete values: 0 (“Off”), 50, 250, 1000 iterations. With Live Preview active, the trainer copies the Gaussian buffer from the GPU into a separate render buffer and triggers a Viewport redraw. With “Off”, the Viewport is only updated after training completes. Performance cost: every 50 iterations ~5–10% slower on M3 Ultra, every 250 iterations ~1–2% slower, every 1000 iterations unmeasurable. Memory overhead constant ~2 GB for the snapshot buffer, independent of the interval. The value serves as the default for new trainings; after training start, the Training Inspector shows the actual live value of the running training. At an interval of 50 the visual impression is a fluid “growing” of the point cloud, at 1000 it looks choppy.

IN PLAIN WORDS

While training is running, you can choose how often the 3D view is updated. “Off” means: no update during training (fastest). “Every 50 Iterations” shows almost in real time how your scene comes together (slightly slower). For comfortable watching of small trainings, “Every 250” is a good compromise.

S6 Throttle Delay

WHERE

Settings → General → Training → Throttle picker. Bound: `AppState.trainingConfig.throttleDelayMs`. Default: `0` (Off).

TECHNICAL

Inserts an artificial delay in milliseconds between training iterations. Four discrete values: 0 (“Off”), 2 (“Light”), 5 (“Moderate”), 10 (“Eco”). Purpose: during longer trainings (several hours) the GPU would otherwise be 100% utilized, which leads to noticeably slower system UI (mouse cursor stutters, other apps become sluggish). The throttle delay gives the GPU pauses in which other tasks can be executed. Performance cost is considerable: with 5 ms throttle a typical 40K training takes about 50–80% longer than without throttle. In performance mode “Eco” (10 ms) the delay per iteration is longer than the iteration itself — factor 2–3× slower. With throttle active, a note appears below the picker: “Throttle is on. Training will be slower than usual.” The app itself does not react noticeably better — only other apps benefit.

IN PLAIN WORDS

If your Mac runs too hot during a long training or other programs become too sluggish, switch on a brake here. “Off” gives the GPU full throttle (fastest). “Light” makes a small pause between every step (slightly slower, but the system reacts better). “Eco” is the strongest brake — good for overnight trainings on a MacBook that should not get too hot.

AI Helpers Tab

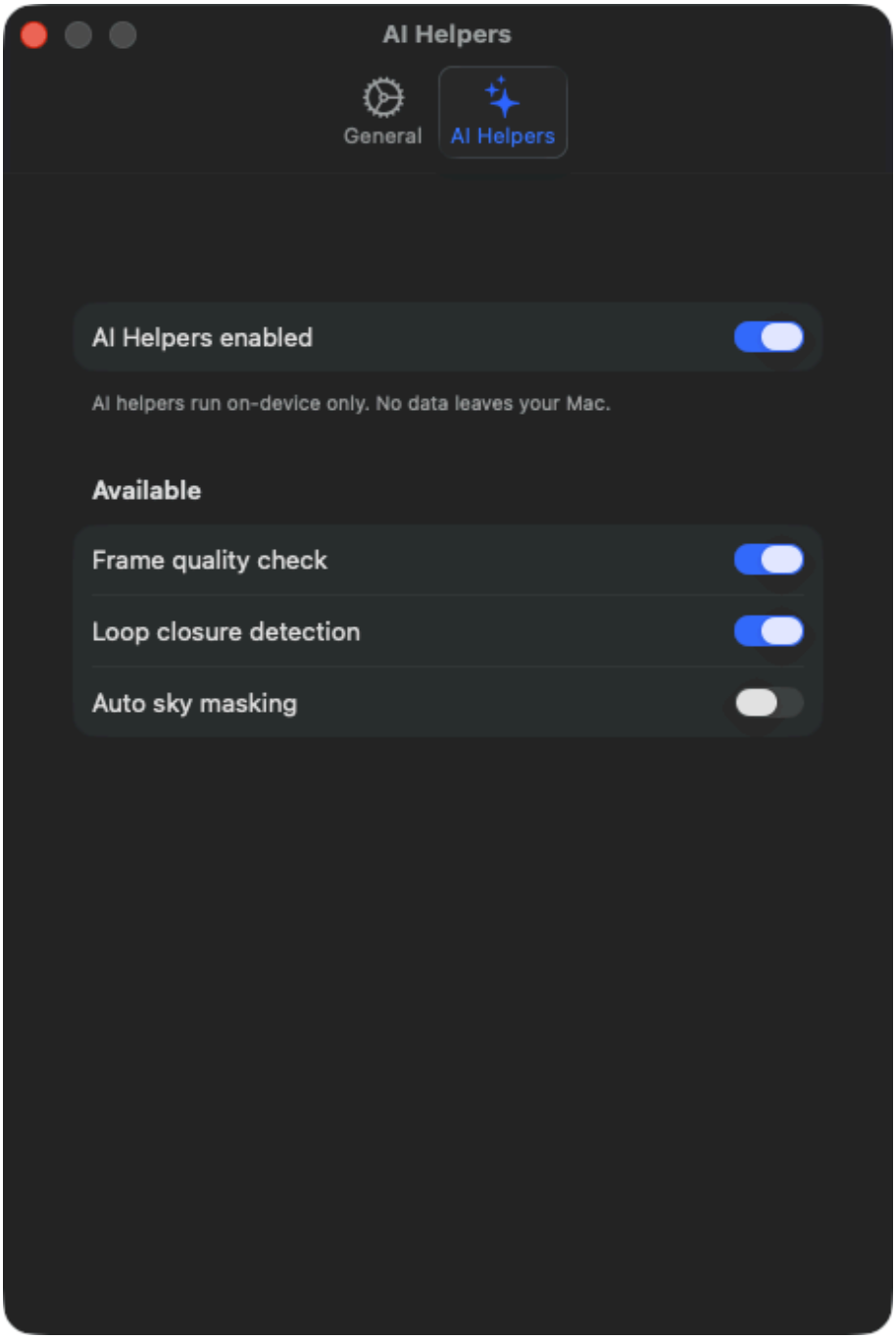

Figure 16: Settings → AI Helpers tab with master switch and sub-toggles

S11 AI Helpers enabled (Master)

WHERE

Settings → AI Helpers → first section → Toggle “AI Helpers enabled”. Bound:. Default: `true`.

TECHNICAL

Master switch over all AI Helpers features in the pipeline. When off, the import and SfM pipeline skip all ML-based preprocessing stages completely — no Apple Vision call, no CoreML model load, no NPU wake-up. When on, the individual sub-toggles (S12–S13) are consulted. The value is remembered across restarts. Affects the following stages: (a) frame quality pre-check before SfM (S12), (b) loop closure detection (S13). Important: when off, the two sub-toggles are disabled and visually grayed out. Footer note emphasizes that all AI Helpers run strictly on-device — no image upload, no cloud processing. The privacy guarantee comes from using exclusively the Apple Vision framework (locally on the Neural Engine) and CoreML models that ship directly in the app bundle.

IN PLAIN WORDS

The main switch for all functions that internally use AI/machine learning. Default is “on”, because the helpers save a lot of time without your images leaving the Mac. If you want them completely off (e.g. to save power or because your Mac has no NPU), switch them off here — the two sub-options below then automatically gray out and do nothing more.

S12 Frame quality check

WHERE

Settings → AI Helpers → Available section → Toggle “Frame quality check”. Bound:. Default: `true`.

TECHNICAL

Activates the frame quality screener (Phase 3.11), which analyzes every imported frame before the SfM call. Pipeline steps per frame: (a) Laplacian variance filter from Apple Vision (blur detection — threshold ~150), (b) histogram-based over/under-exposure check (threshold: >5% pixels at 0 or 255), (c) blank frame detect (standard deviation < 5 over all pixels). Frames that pass all three checks go through directly. Frames that fail at least one check trigger a modal confirmation dialog that lists every problematic frame with thumbnail and reason and asks whether it should be removed. Important: no automatic deletion — the dialog is always required, the user keeps the final decision. Performance: ~50 ms per frame on M3 Ultra, runs in parallel. When off, all frames are forwarded to SfM unchecked. With the master (S11) disabled, this toggle is visually grayed out and has no effect. Shipped status per memory: SHIPPED 2026-05-23.

IN PLAIN WORDS

Before the actual training, the app looks at every photo: is it blurry? completely dark or white? blank? If so, it asks you whether you want to throw the image out — it never removes anything automatically. This saves many hours later, because a single totally blurry image can sometimes ruin the whole training. Default is “on”, because the cost is almost zero and the benefit is large.

S13 Loop closure detection

WHERE

Settings → AI Helpers → Available section → Toggle “Loop closure detection”. Bound:. Default: `true`.

TECHNICAL

Activates the Apple Vision feature-print-based loop closure detection. For every imported frame, a ~768-dimensional feature vector is computed, which represents a neural embedding of the image content. All feature prints are then compared pairwise via cosine similarity. Pairs with similarity > 0.85 and frame-index distance > 50 (i.e. non-adjacent frames) are identified as “loop closure candidates” and written to a sidecar JSONL file in the project folder. Informational only — the imported image sequence is not modified. Purpose: gives the SfM solver (especially COLMAP) a hint that these frames cluster together in 3D space. For native SfM the sidecar information is currently only documentary; COLMAP uses the hints internally via custom matches file (manual integration possible, not automatically wired). Performance: ~200 ms per frame on M3 Ultra, runs in parallel. When off, no feature prints are generated. With the master (S11) disabled, visually grayed out.

IN PLAIN WORDS

When you walk around an object while shooting and end up at the starting point again, it helps the computer enormously to know that. This option automatically detects which of your photos were taken “from almost the same spot” and writes that into a small helper file. SfM tools (especially COLMAP) can use this information to deliver a cleaner 3D reconstruction. Default is “on”, because it runs without your intervention and changes nothing in your images.

Inspector-Mirror Settings

The remaining settings entries (S17–S33) from the inventory table are mirrors from the Expert Inspector and are documented in Chapter 2 (Inspector controls I12–I29). They do not appear physically in the Settings window, but were listed in the inventory only because they run via `TrainingConfig` properties that are persisted via and therefore formally have settings character. For substantive explanations see there.

When What?

Setting	Scope	Persistence
S1 Default Mode	App-Global	App Restart
S2 Language	App-Global	App Restart
S3 Viewport Background	App-Global (Default) + Runtime	App Restart
S4 Auto-Rotate After Training	App-Global	App Restart
S5 Live Preview Interval	Default for new trainings	App Restart
S6 Throttle Delay	Default for new trainings	App Restart
S11 AI Helpers Master	App-Global	App Restart
S12 Frame quality check	App-Global	App Restart
S13 Loop closure detection	App-Global	App Restart

App-Global = affects all projects. Default for new trainings = affects only the next training created; running sessions remain unchanged. Current training = takes effect immediately on the running training configuration, but does not persist without explicit re-import.

CHAPTER

Chapter 4 — Auxiliary Windows

Besides the main window (3D viewport plus Inspector), RadianceKit manages seven additional windows, all of which are opened via the Help menu. The list from top to bottom: User Guide (⌘?), Keyboard Shortcuts (⌘/), Open Training Logs… (does not open an app window, but the Finder; therefore not covered further here), Manage Storage…, Pareto Dashboard… (⇧⌘D), Holdout Analysis… (⇧⌘H), BayesOpt Console… (⇧⌘B). Three of these — Dashboard, Holdout, BayesOpt — are standalone analysis tools. They each have their own view-model stack, read or write JSON files on disk, and there is a CLI argument for each that lets you point the window at a specific file directly at app startup (`--dashboard-dir` , `--holdout-file` , `--bayesopt-autorun`).

The four simple windows (User Guide, Keyboard Shortcuts, Manage Storage, plus the submenu items Open Training Logs / Open Exports Folder) get a short entry per control. The three analysis windows are documented in more detail — each with an introduction explaining what you see in the window, when you should open it, and how to interpret the picture shown.

At the end of the chapter there is a cross-reference section to the main window's Inspector: what you can meaningfully read from the live loss chart and the Gaussian count display during a running training.

User Guide (W1–W4)

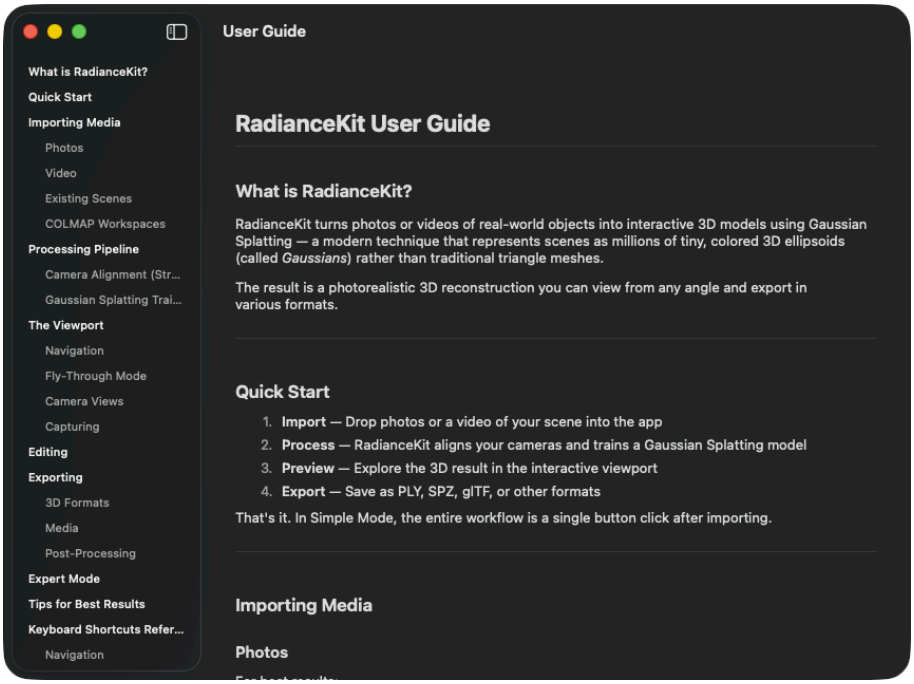

Figure 17: User Guide window with sidebar on the left and rendered Markdown content on the right

What it is: A built-in help window that renders the `guide_<language>.md` shipped with the app. The language is derived from the Settings (General tab → Language) or, if "System" is set there, from the macOS language preferences. Layout is classic: sidebar with all headings on the left, body text on the right.

WHEN TO OPEN When you need a quick reminder of a single point — so as a keyword substitute. The full reference is this manual; the built-in help window is more like what a `--help` would be on the command line. It is updated with every app release, but is intentionally kept more superficial in content.

W1 NavigationSplitView (Sidebar + Detail)

WHERE

Help → User Guide (⌘?)..

TECHNICAL

Two-column layout with a narrow sidebar (at least 180 pt wide) for the content tree and a scrollable detail area for the actual Markdown content. The window has a minimum size of 700 × 500 pt. On first open, the window loads the matching `guide_<lang>.md` from the app bundle (fallback `guide_en.md`), parses it into block records (headings H1–H4, paragraphs, lists, tables, separators) and separately extracts the heading structure for the sidebar. Inline formatting (bold, italic, codespan) is rendered via the built-in Markdown engine. The language is read from the app settings, with the special case of Chinese (`zh-Hans`) and Brazilian Portuguese (`pt-BR`), which are kept as full locale tags because these variants differ from zh and pt respectively.

IN PLAIN WORDS

The built-in help text, with the topic list on the left and the content on the right. The language adjusts automatically based on your system settings. Works offline, but is intentionally only a short version — the full reference is this manual.

W2 List (Heading sidebar)

WHERE

Left column in the User Guide window..

TECHNICAL

List of all H2 and H3 headings of the current Markdown document. H2 entries appear without indentation in Medium font weight, H3 entries with 16 pt left indentation and a reduced foreground style. H4 and deeper are ignored, because the depth would otherwise make the sidebar cluttered. Anchor IDs are generated from the heading text via slugification (lowercase + spaces to dashes + filtering for letters/numbers/dashes — the same algorithm GitHub uses for its Markdown anchors, so external URLs to the docs would potentially land on the same anchor as well). The list uses the native macOS style.

IN PLAIN WORDS

The navigation bar on the left side. Tap an entry and you jump to the section.

W3 Button (Heading → anchor jump)

WHERE

One button per sidebar row..

TECHNICAL

Each sidebar entry is a button that sets the current anchor, but visually looks like a list entry. An observer variable then triggers the scroll jump to the matching anchor with a smooth animation over 0.3 s. After the jump, the anchor value is reset so that the next click on the same anchor fires again (otherwise the observer wouldn't trigger again because the value hasn't changed).

IN PLAIN WORDS

Clicking takes you to the corresponding spot in the text on the right.

W4 ScrollView (detail content)

WHERE

Right column..

TECHNICAL

Scrollable, vertically stacking content area with lazy rendering, because longer guides can easily have over 200 Markdown blocks — a non-lazy variant would instantiate them all at once. Each block gets its own ID, either the heading anchor (for jumpable H1–H3) or an index placeholder. Maximum width is 720 pt, padding 32 horizontal / 24 vertical, so long lines retain a well-readable layout. Tables are rendered cell-by-cell with horizontal stacks and separators; inline code via the built-in Markdown engine. Real code blocks are currently treated as paragraphs — a known limitation of the Help window.

IN PLAIN WORDS

The actual help text. Scrollable, well-readable width, clear typography.

Keyboard Shortcuts (W5–W6)

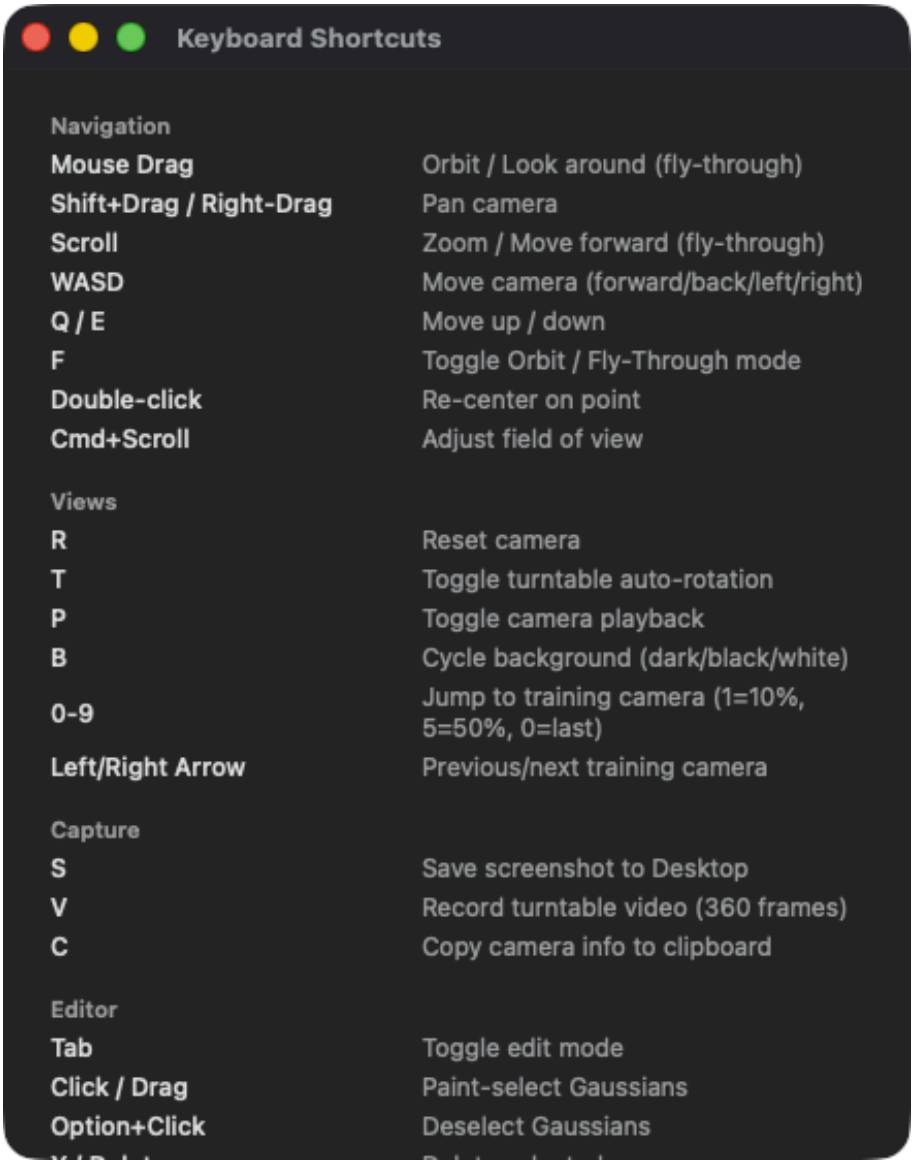

Figure 18: Keyboard Shortcuts window — five groups Navigation/Views/Capture/Editor/Training with hotkey column on the left and description on the right

WHAT'S IN THE IMAGE Static reference list in five sections. **Navigation:** Mouse Drag (Orbit/Fly), Shift+Drag/Right-Drag (Pan), Scroll (Zoom), WASD (fly-through movement), Q/E (Up/Down), F (Toggle Orbit/Fly), Double-click (Re-center), Cmd+Scroll (FoV adjust). **Views:** R (Reset Camera), T (Auto-Rotation), P (Camera Playback), B (Background cycle), 0–9 (jump to training cam 1=10%/5=50%/0=last), Left/Right Arrow (Prev/Next Cam). **Capture:** S (Screenshot to Desktop), V (Turntable Video), C (Copy Camera Info). **Editor:** Tab (Edit mode), Click/Drag (Paint-Select), Option+Click (Deselect), X / Delete (delete selection), Cmd-Z (undo last deletion), [/] (brush size smaller/larger), Esc (clear selection). **Training:** Start, Pause/Resume, Cancel, Continue +5K/+10K/+20K via the menu shortcuts in M9–M14.

What it is: A simple static overview of all keyboard shortcuts — Navigation, Views, Capture, Editor, Training. Content is hardcoded in, no Markdown loading.

WHEN TO OPEN When you're looking for the fastest way to do something in the viewport. WASD fly-through, R for camera reset, B for background cycling — they're all here.

W5 ScrollView (content area)

WHERE

Help → Keyboard Shortcuts (⌘/)..

TECHNICAL

A simple scroll area with a vertical list inside. Padding 20 all around, no sidebar navigation tree (the list is short enough). Content is grouped into five sections (Navigation, Views, Capture, Editor, Training). One row per key combination with translatable text in both columns. Left column (key code) fixed to 180 pt width, so the descriptions on the right stay vertically aligned. No interaction except scrolling — clicking on a row doesn't trigger anything; the shortcuts are real keyboard modifiers in the menu and on the viewport.

IN PLAIN WORDS

Table of all shortcut keys. Static cheat sheet for quick lookup.

W6 VStack (shortcut sections)

WHERE

Inside the ScrollView..

TECHNICAL

Left-aligned stacked sections with 16 pt spacing. Within the five sections, each contains a heading + row sequence. Headings use a secondary subheadline style — intentionally not a title format, because the sections don't need to be navigable. Content is intentionally flat (no disclosure, no search, no filter) so that the component runs unchanged on every macOS version and the file stays readable.

IN PLAIN WORDS

The grouping of keys by function (Navigation, Views, Editor, and so on).

Manage Storage (W7–W12)

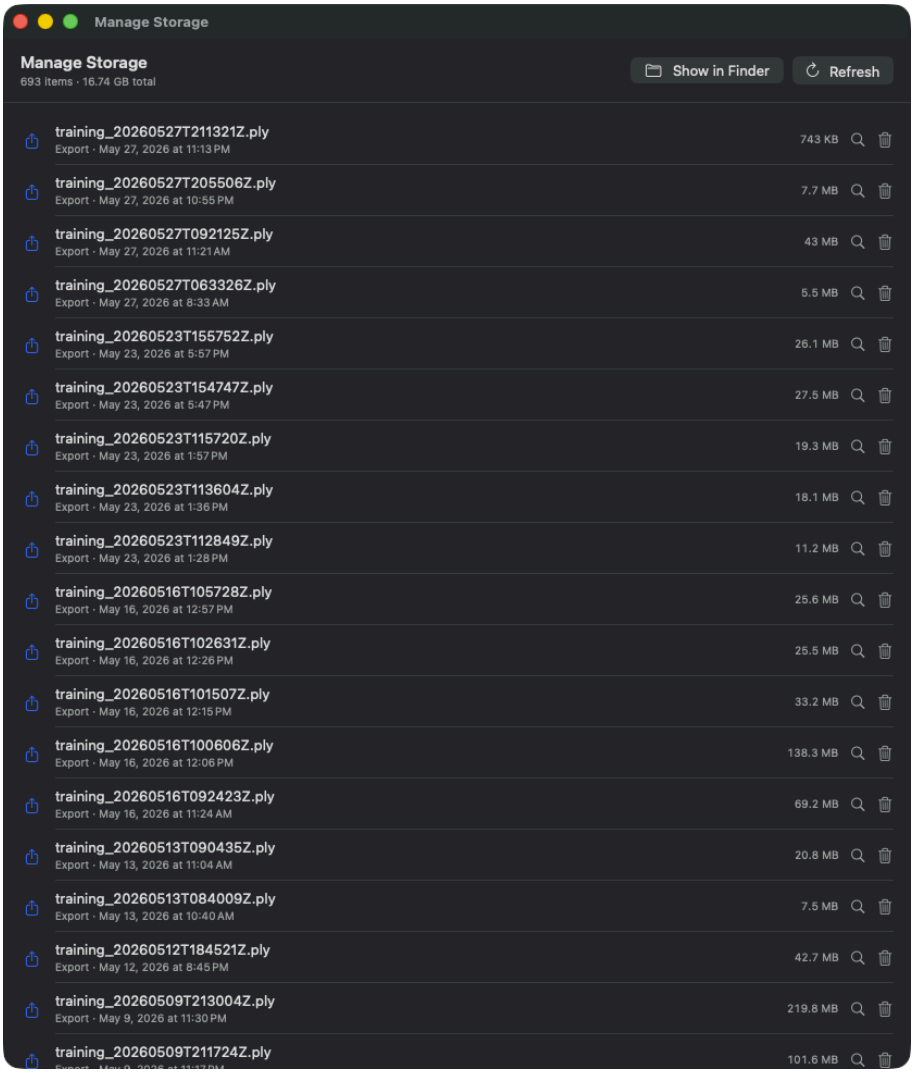

Figure 19: Manage Storage window — header shows „693 items · 16.74 GB total", table with Export PLY files sorted by date, each with format pill + filename + size + date

WHAT'S IN THE IMAGE Table view of all files managed by RadianceKit. Header counts 693 items, 16.74 GB total size. Toolbar at top: "Show in Finder" + "Refresh". Each row: PLY icon, filename (e.g. `training_20260527T211321Z.ply`), export date, size (varies 7 KB to 218 MB), magnifier icon (Reveal) and trash icon (Move to Trash). Files are sorted by date, newest at the top. In this demo recording PLY exports dominate because a lot of work was done with `--benchmark`.

What it is: A disk usage overview of everything RadianceKit stores under `~/Documents/RadianceKit/` — Logs, Exports, Scenes, Capture bundles (from the iOS companion), Imports (staging copies of the input images). One size in bytes per entry and two buttons: "Show in Finder" and "Move to Trash". This is NOT automatic cleanup — the app doesn't delete anything itself; you decide per entry.

WHEN TO OPEN When the disk is filling up. Logs in particular accumulate (one JSONL per training attempt, plus the `_qualityMetrics.json`); exports too of course (PLY 100% raw data, one per export). Also useful after a crash when the imports staging directory

still has old copies of the input images lying around (see “Disk pressure incident” in `dev_v549f-needle-reduction.md`).

W7 Button “Show in Finder”

WHERE

Top right of the header in the storage browser window..

TECHNICAL

Opens the entire RadianceKit directory (`~/Documents/RadianceKit/`) in the Finder, so you can see the folder structure directly and also manipulate it with the Finder itself. The action opens a new Finder window and does not switch into the app sandbox container — `~/Documents/RadianceKit/` is the regular Documents domain accessible to apps, not a sandboxed container path.

IN PLAIN WORDS

Opens the directory in the Finder so you can work with the files yourself.

W8 Button “Refresh”

WHERE

Header, next to the Finder button..

TECHNICAL

Triggers a background scan that runs on a user-initiated asynchronous task so that scanning large directory trees doesn’t block the UI. The actual walk goes through every known subfolder (Logs, Exports, Scenes, Captures, Imports) and produces one storage entry per direct child. Per entry the recursive size is determined — preferably the actual disk usage (including APFS hardlink sharing) with fallback to the logical file size.

IN PLAIN WORDS

Re-reads the list in case you deleted or added something in the Finder in between.

W9 List (storage entries)

WHERE

Main content below the header..

TECHNICAL

List with this layout per row: category- specific SF Symbols icon (document for Logs, upload arrow for Exports, cube for Scenes, tray for Imports), name + subtitle (Kind label + formatted modification date), bytes counter on the right (right-aligned, monospaced), Reveal button (magnifier icon), Trash button (trash can). Sorting: primarily by Kind (Scenes first, then Exports, Logs, Captures, Imports, Other), secondarily by modification date descending (newest at top). If the scan is still running, the area shows a “Scanning…” progress indicator instead. If nothing was found, an empty-state display with a tray icon.

IN PLAIN WORDS

List of all your RadianceKit data, sorted by type and recency. Per entry you see the size and can delete directly.

W10 Row button “Reveal in Finder”

WHERE

Per row, magnifier icon on the right..

TECHNICAL

Opens the Finder and selects the specific item (file or folder). Difference from W7: W7 opens the root directory; W10 highlights exactly this one entry. Practical workflow: identify a large entry, click on the magnifier, then copy it to an external volume for example.

IN PLAIN WORDS

Jumps to this entry directly in the Finder so you can find it quickly.

W11 Row button “Move to Trash”

WHERE

Per row, trash icon on the right next to the magnifier..

TECHNICAL

Triggers the confirmation dialog box (W12). Only after confirmation does the macOS standard “move to trash” operation run (so reversible, no direct deletion). After successful trash, the entry is removed from the list and the total byte counter is updated. On errors, a modal error dialog is displayed.

IN PLAIN WORDS

Moves the entry to the trash. A dialog asks first.

W12 ConfirmationDialog (delete confirmation)

WHERE

Triggered by W11, displayed as a macOS sheet..

TECHNICAL

Standard confirmation dialog with a dynamic title “Delete <name>?” and a message line that explicitly points out that the entry goes to the trash and can be restored from there (until the trash is emptied). Two buttons: “Move to Trash” as destructive action (shown in red) and “Cancel” with automatic Esc binding. The dialog is non-modal in the sense that it only blocks this window, not the whole app — that’s macOS standard for reversible deletions.

IN PLAIN WORDS

Safety prompt before deletion. “Move to Trash” is reversible — as long as the trash isn’t emptied.

Pareto Dashboard (W13–W22)

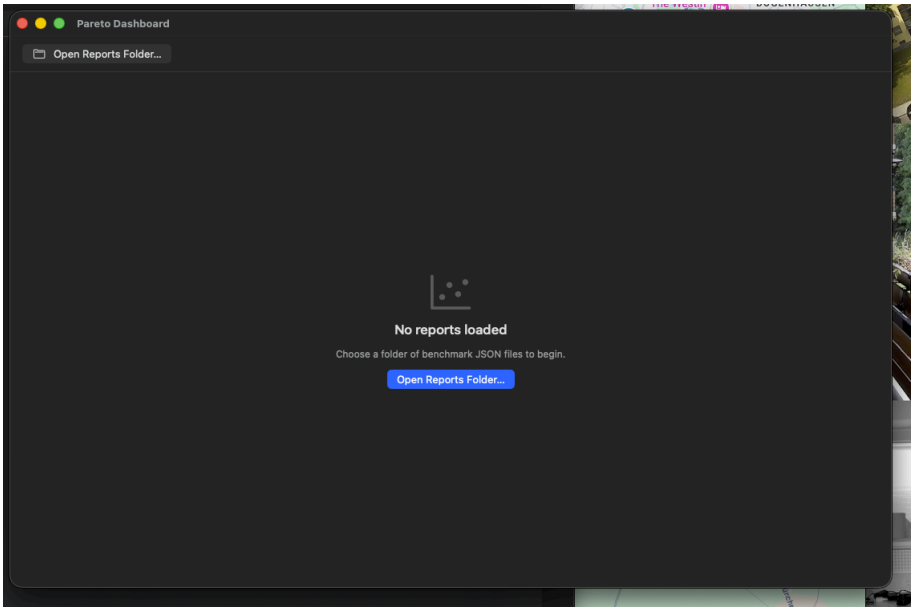

Figure 20: Pareto Dashboard — empty state before report import

Empty state (after first open) — empty state with call-to-action “Open Reports Folder...”. The data points appear as soon as training reports are loaded, see next shot.

Figure 21: Pareto Dashboard with 129 loaded benchmark reports — Gaussians vs PSNR with Pareto front, Scene/Strategy/Mip filter

WHAT'S IN THE IMAGE Header toolbar shows “129 reports of 129” (all reports in the selected folder parsed successfully — 21 additional files could not be parsed due to older format, see hint list on the right). Axes: X-axis picker on `Gaussians`, Y-axis picker on `PSNR (dB)`. Scatter plot: green points = Classic strategy, blue points = MCMC. The dashed Pareto front line runs along the best-achieved PSNR values and plateaus around PSNR≈30 dB from about 500K Gaussians onwards. Filter chips on the right: 7 scenes

(bicycle, bonsai, family, flowers, kitchen, stump, truck), 2 strategies (classic, mcmc), 3 Mip-Splatting options (All, On, Off). Currently all filters are open, hence the dense point cluster.

What it is: A multi-run comparison tool. In the past you have trained several scenes, or the same scene with different presets — each of these training runs produces (if you passed `--benchmark` or called via the Benchmark function) a JSON report file containing among other things final PSNR, SSIM, LPIPS, Gaussian count and wallclock time. The Dashboard reads a whole folder of such reports at once and plots them as a 2D scatter with selectable axes. Additionally, the Pareto front (the set of non-dominated points) is drawn as a dashed line.

WHEN TO OPEN After you have produced at least three or four training reports. With fewer points the frontier line is not meaningful. Typical use case: you tried to reconstruct an outdoor scene and successively ran P3 Balanced (Classic), P4 Quality (Classic), P7 MCMC Quality and P9 Outdoor (tuned) — now you want to know which configuration delivers the best PSNR per second of training time, or which requires the fewest Gaussians for a given PSNR.

HOW TO INTERPRET Both axes are freely selectable (X-axis:,, `psnr`, `ssim`, `lpips`, …; Y-axis the same). The Pareto front logic in `ParetoFront2D.indices` knows for each metric whether “smaller = better” (e.g. LPIPS, Loss, Time) or “larger = better” (PSNR, SSIM) — so depending on the axis choice, the line runs from bottom-left to top-right or from top-left to bottom-right, always along the best achieved combination. A point is Pareto-optimal if NO other point is at least as good in BOTH dimensions (so no other point dominates it). Pareto-optimal points lie on the line, other points to the right/above (depending on axis orientation). Points ON the line are the real candidates for “best preset”; points FAR from the line are wasted training time.

FILTER CHIPS You can restrict the selection to a particular scene (if you only want to compare outdoor runs, e.g.), to a particular strategy (Classic or MCMC), or to Mip-Splatting on/off (relevant after Phase Q1.5, where Mip remains as an opt-in advanced flag).

You have three reports for the “truck” scene under `~/Documents/RadianceKit/Reports/`: Run A (P4 Quality, 40K iter, 524K Gs, 105 s, PSNR 23.4), Run B (P7 MCMC, 200K iter, 150K Gs, 693 s, PSNR 24.6), Run C (P9 Outdoor, 100K iter, 1.25M Gs, 312 s, PSNR 25.8). Set X-axis to trainingTime, Y-axis to PSNR. Run B lies upper right, Run C even further upper right, Run A lower left. The Pareto front connects A and C — both non-dominated. Run B is “lost” (C is better in Time AND PSNR). Insight: for “truck” the MCMC default isn’t worth it; either fast+ok (A) or long+very good (C). Save the configuration from C as your own preset (Inspector → I1 Save Preset).

Next action: Save the best configuration as a preset. Concretely: look at the Pareto points (hover shows PSNR/SSIM/LPIPS/Gs/Time in the tooltip), decide which one suits you best in the time-vs-quality trade-off, open the corresponding report (filename contains run timestamp), copy its training configuration into a new run, or save it after the next training session as a preset via the Inspector.

W13 Button “Open Reports Folder…”

WHERE

Toolbar top left..

TECHNICAL

Opens a folder picker dialog with the prompt “Select a folder containing benchmark .json reports”. After confirmation, a background task runs that parses all `.json` files in the folder sequentially. Faulty reports (broken JSON, wrong schema) are collected and shown at the bottom of the sidebar as “N file failed to parse” — no crash. If a second click happens while a first load is still running, the previous task is canceled so that two results don’t write into state simultaneously.

Also via CLI: `--dashboard-dir /path/to/reports` loads the folder directly at app startup.

IN PLAIN WORDS

Selects the folder where your benchmark reports live. Default path is `~/Documents/RadianceKit/Reports/`. Then loads all JSONs at once.

W14 Picker “X-Axis”

WHERE

Above the chart, on the left..

TECHNICAL

Menu picker with all available metric axes of the dashboard module (PSNR, SSIM, LPIPS, Gaussian count, training time and so on). Default is Gaussian count. On change, the hovered point is reset because a previously highlighted position in the old axis coordinate system no longer makes sense after axis change. The picker is constrained to content width so it doesn’t span the entire width.

IN PLAIN WORDS

Which metric should be on the horizontal axis. Usually “training time” or “Gaussian count” because those are the “costs” you want to compare.

W15 Picker “Y-Axis”

WHERE

Above the chart, next to X-Axis..

TECHNICAL

Identical to W14, except the default is PSNR. The axis choice is stored independently, so the user can also pick nonsense combinations (X=PSNR, Y=PSNR — would throw all points onto a diagonal). Such combinations are not caught, however; a deliberate decision, because a comparison “SSIM vs PSNR” is quite interesting for seeing how consistently the metrics behave.

IN PLAIN WORDS

What’s on the vertical axis. Normally “PSNR” or “SSIM” as a quality measure.

W16 Toggle “Show Pareto Front”

WHERE

To the right of the axis pickers..

TECHNICAL

Standard macOS toggle. If active, a line with the computed 2D Pareto front is drawn in the Pareto chart in addition to the point cloud. Style: dashed (dash pattern 4–4), gray semi-transparent, line width 1.5 pt. The Pareto computation runs on the main thread — with the typical number of reports (≤ ~50) this is fast without issue. If the toggle is off, the line is omitted so only the bare points remain.

IN PLAIN WORDS

Shows the line running through the “best-so-far” points. If the line is in the way (e.g. because you only want to compare individual trades), turn it off.

W17 Chips “Scene” filter

WHERE

Right sidebar in the Dashboard window..

TECHNICAL

Filter chips for every scene appearing in the loaded reports. Custom flow layout that automatically wraps chips into multiple lines as soon as the width is exhausted. Active chips get the accent background, inactive ones a neutral standard material background. Multi-selection is possible (set semantics); if no chip is selected, all scenes count as “passed through” — i.e. the set logic is “empty selection = all”, not “empty selection = nothing”.

IN PLAIN WORDS

Clicking a scene name filters the points to only this scene. Multi-selection possible. Empty = all scenes.

W18 Chips “Strategy” filter

WHERE

Below Scene filter in the sidebar..

TECHNICAL

Exactly like W17, but for training strategies — typically the two values “classic” and “mcmc”, derived from the strategy field of the benchmark report JSONs. Helpful if you have mixed reports of both strategies and only want to see one kind (e.g. “only show MCMC runs because I’ve already excluded Classic”).

IN PLAIN WORDS

Filter by Classic or MCMC. By default both are active.

W19 Chips “Mip-Splatting” filter

WHERE

Below Strategy filter in the sidebar..

TECHNICAL

Three-valued filter (instead of a set like W17/W18): “All” / “On” / “Off”. Background: Mip-Splatting was evaluated in Phase Q1.5 as an experimental multi-scale improvement and the final verdict was “no nice win across the board; keep as opt-in flag”. When you do Mip on/off comparisons you often want to separate very sharply. Hence the dedicated ternary filter with the states “let everything through”, “only Mip on”, “only Mip off”. The sidebar section is only displayed if at least one Mip report AND at least one non-Mip report is in the data set (otherwise filtering makes no sense).

IN PLAIN WORDS

If you want to compare Mip-Splatting on/off, here is a three-way filter. Otherwise ignore.

W20 ChipButton (filter toggle, all/on/off)

WHERE

Helper component, used in W17/W18/W19..

TECHNICAL

Minimalist button wrapper. Content: label text with caption font size and padding 10 horizontal / 5 vertical. Background conditional: if active → app accent color with white text; otherwise neutral standard material background with black text. Shape is a capsule (pill-like). Plain button style so the capsule material isn’t overlaid by a system border.

IN PLAIN WORDS

The round filter buttons themselves. Visually like an iOS tag.

W21 Chart (Pareto scatter)

WHERE

Center area of the Dashboard..

TECHNICAL

Swift Charts diagram with two layers: 1. one point per report — position from the chosen X and Y metrics, color by strategy, symbol by Mip status. Symbol size normal 80, highlighted 200 (if the ID matches the currently hovered report).
2. a line for the Pareto front, only if the toggle is on.

Chart overlay: a transparent rectangle registers mouse motion; per frame the Euclidean nearest point position in the plot frame is determined and the hovered report is updated if the distance is under 24 px (otherwise reset). So you get the tooltip without clicking — hovering is enough.

IN PLAIN WORDS

The actual scatter plot. Each point is a training run. Hover for detail tooltip.

W22 Tooltip (hover detail)

WHERE

Below the chart, displayed on hover..

TECHNICAL

Horizontal stack: scene name (headline), strategy tag (caption), separator, then PSNR/SSIM/LPIPS/Gs/Time metrics each in a small vertical group (label + monospaced value). If Mip was activated, additionally a “Mip” capsule tag in accent color. Background semi-transparent blur, rounded rectangle with 8 pt radius. Only displayed when the mouse is actually over a point. Disappears automatically on leaving.

IN PLAIN WORDS

The detail card at the bottom when you hover over a point with the mouse. Shows all quality metrics and the run configuration at once.

Holdout Analysis (W23–W29)

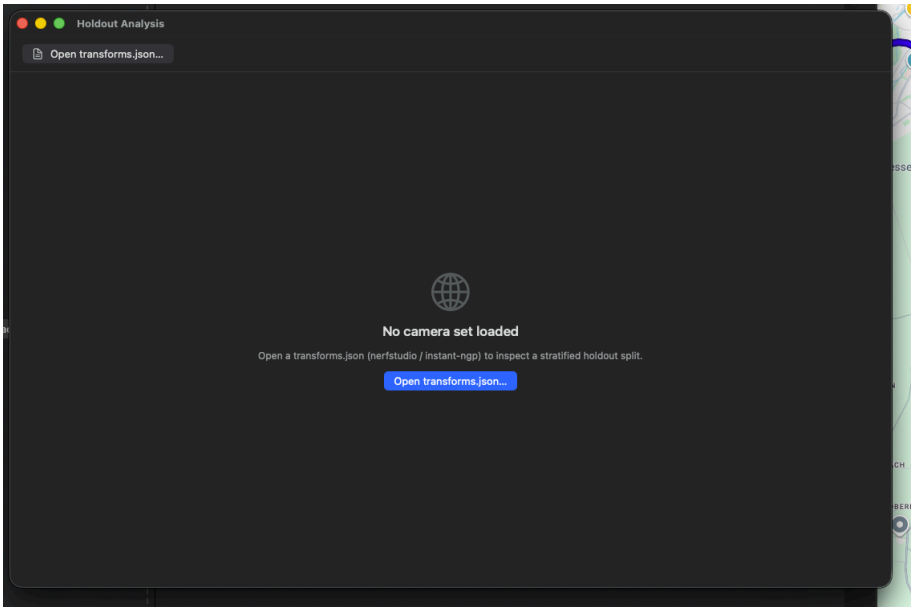

Figure 22: Holdout Analysis — empty state before loading a transforms.json

Empty state with empty state and call-to-action “Open transforms.json...”. Accepts NeRF Studio and Instant-NGP format.

Empty state (after first open) — the camera markers appear as soon as a `transforms.json` is loaded, see next shot.

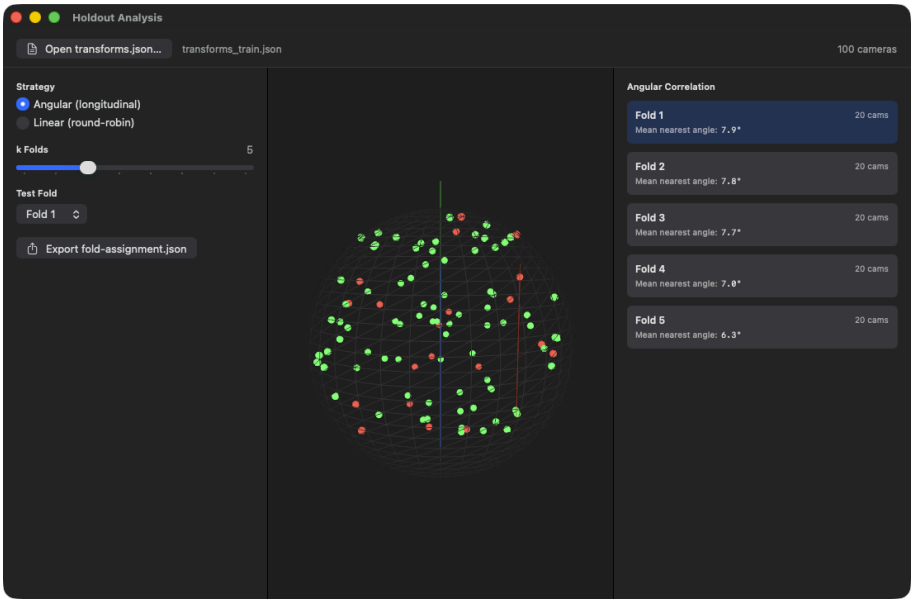

Figure 23: Holdout globe with 100 NeRF Blender mic cameras, 5 folds of 20 cameras each, angular strategy active

WHAT'S IN THE IMAGE Header shows loaded file (`transforms_train.json`) and cam count (“100 cameras”). Left sidebar: strategy picker with two options — `Angular (longitudinal)` active (aligns folds along longitudinal/latitudinal sectors on the sphere so each test fold is geometrically dense) vs `Linear`

`(round-robin)` (order-based, every k-th frame as the test set). k-folds slider is on 5, test-fold picker on Fold 1. Export button produces a `fold-assignment.json` for Nerfstudio/Instant-NGP. Middle panel: 3D globe projection of all 100 cameras — green points = Train, red points = current test fold (Fold 1 with 20 cameras). Right sidebar (Angular Correlation): per fold 20 cams + Mean Nearest Angle (Fold 1: 7.9°, Fold 2: 7.8°, Fold 3: 7.7°, Fold 4: 7.0°, Fold 5: 6.3°) — a smaller value means the cameras within this fold lie close together, so the Holdout split is spatially coherent.

What it is: A 3D visualizer for your camera arrangement with cross-validation logic. You load a `transforms.json` (the standard format from Nerfstudio / Instant-NGP for camera poses), the app reads all cameras, projects their view directions onto a unit sphere and shows them as small sphere markers on a virtual globe. Then it divides the cameras into `k` folds (per the selected strategy: angular or linear), marks the training portion green and the test portion red (Holdout), and computes per fold an Angular Correlation score that tells you how far away the test fold is from the training fold in view-angle space.

WHEN TO OPEN When you want to do Holdout evaluation — i.e. how well does your model generalize to unseen viewpoints? Default in training is "every-8th view as Holdout" (Mip-NeRF360 convention), but that is a very linear split. If your images are clustered in time for example (one side of the object first, then the other), then "every-8th" is not representative — a random sequence position ends up in the test, but all its neighbors are in training, which is too easy. With "angular" you stratify across the view-angle space instead: each fold contains cameras from all areas of the orbit so the test really probes generalization gaps.

HOW TO INTERPRET Angular vs Linear: - Angular (default): divides the cameras by longitudinal angle (ϕ coordinate around the Y axis) into `k` equal sectors. Fold 0 contains cameras with ϕ ∈ [0°, 360/k°), Fold 1 the next ones, and so on. Advantage: each fold covers a portion of the orbit; the test fold is spatially compact but broadly distributed across the world dataset. Good for classic orbit recordings. - Linear (round-robin): Fold index = (image_index modulo k). That is the simple "every k-th" split. Works if the image order has NO spatial bias (e.g. randomly sorted drone shots). Works poorly if the images cluster in time.

In the 3D globe you immediately see: green points (training) and red points (test). If the red points all cluster in one corner, the Holdout is poor (no good generalization test). If they lie evenly between the green ones, it's good. The Angular Correlation score per fold (right sidebar, in degrees) additionally says: smaller value = the test is close to training (each test camera has a nearby training camera, easy test); larger value = the test is far from training (harder generalization).

You captured your truck scene with 251 images, export via menu item M33 (Export SfM transforms.json) a Nerfstudio file. Open the Holdout window (⇧⌘H), load the JSON via "Open transforms.json…", look at the globe. k=5 (default) gives you 5 folds. Click on "Fold 3" — check whether the red markers are reasonably even. If yes: "Export fold-assignment.json", put the exported file in the reports folder, and at the next training run with `--benchmark` (or the corresponding Inspector settings) exactly this fold assignment is used as the test Holdout — instead of the default "every-8th".

W23 Button “Open transforms.json…”

WHERE

Toolbar top left..

TECHNICAL

Opens a file picker dialog restricted to JSON files. After confirmation the Holdout module loads the file. The loader parses both the Nerfstudio format (camera intrinsics plus list of frames with image path and transform matrix) and the Instant-NGP format (same structure). For each frame the view direction is extracted from the transform matrix (z-axis of the camera local basis) and stored. If parsing fails, an error message is shown in the status area.

Also via CLI: `--holdout-file /path/to/transforms.json` opens the window directly with the file loaded.

IN PLAIN WORDS

Loads your camera poses JSON. Standard are Nerfstudio and Instant-NGP exports. RadianceKit itself can export transforms.json via Menu → Export → SfM.

W24 Picker “Strategy” (angular/linear)

WHERE

Left sidebar, at the top..

TECHNICAL

Radio picker with two options: Angular and Linear. Strategy change automatically triggers a recomputation of the folds. The view directions are a list of 3D unit vectors on the sphere; the angular strategy projects them onto the longitudinal angle ϕ and sorts, the linear strategy simply does a modulo split over the frame index.

IN PLAIN WORDS

Angular for evenly distributed orbit shots (default, safe), Linear only if your images don’t cluster spatially.

W25 Slider “k Folds”

WHERE

Left sidebar, in the middle..

TECHNICAL

Slider from 3 to 10, step size 1. On change, the fold computation is automatically restarted so that the folds list, the training/test indices and the per-fold score are immediately recomputed. The selected value is displayed as monospaced-digit text next to the label on the right.

Rule of thumb: k=5 is standard (gives you 20% test per fold, which is common for cross-validation). k=10 if you have a lot of data and need more folds for statistical significance. k=3 if you have little data.

IN PLAIN WORDS

How many folds in the split. 5 is the default and fits almost always.

W26 Picker “Test Fold”

WHERE

Left sidebar, below the k slider..

TECHNICAL

Menu picker. Options are dynamically 0..<k, labels “Fold 1” through “Fold N” (so 1-indexed in the UI, 0-indexed internally). If the previously selected index is ≥ k (e.g. because you reduced k from 10 to 5), it is automatically reset to 0. The selected test fold is shown in red on the globe, all others in green.

IN PLAIN WORDS

Which fold is currently the test fold. You can click through and see how each individual fold looks on the globe.

W27 Button “Export fold-assignment.json”

WHERE

Left sidebar, at the bottom..

TECHNICAL

Opens a save dialog with default filename `fold-assignment.json`. After confirmation the Holdout module encodes the current split into a JSON schema (per-frame fold assignment plus strategy meta block). This file can then be passed to the next training run with `--benchmark`, so the same Holdout is used for the final metric evaluation. Write errors are shown as error text; success in green text as “Saved to (filename)”.

IN PLAIN WORDS

Saves the current train/test split as JSON. You can then pass this file directly to the training so the same test set is used again.

W28 SCNView (3D Camera Globe)

WHERE

Center panel in the Holdout window..

TECHNICAL

SceneKit globe view. The scene consists of: a wireframe sphere (radius 1.0, 36 segments, dark gray), three colored axis stubs (red/green/blue for X/Y/Z, each 1.2 long), and per camera a small marker sphere (radius 0.03) at the corresponding view direction position on the unit sphere (slightly outside so it doesn’t disappear INTO the wireframe sphere). The markers are NOT rebuilt on each fold change — rebuild is only needed when the frame list changes (i.e. a new JSON is loaded). Instead, per update an in-place update of the material colors runs: red for test indices, green for training, light gray if neither. So slider ticks stay performant even at N > 1000 cameras.

Camera control is enabled — you can rotate, zoom and pan the globe with the mouse. Lighting makes sure the markers don’t look flat. Background is dark gray.

IN PLAIN WORDS

The 3D globe with the camera positions. Green = training, red = test, light gray = unassigned (doesn’t occur, all cameras belong somewhere). With the mouse you can rotate and zoom the globe.

W29 FoldCard (tap to select fold)

WHERE

Right sidebar, “Angular Correlation” section..

TECHNICAL

One card view per fold — rounded rectangle with 6 pt radius, padding 10, vertical layout with two rows (top “Fold N” + camera count, bottom “Mean nearest angle:” + value in degrees). Background color conditional: active fold = accent color semi-transparent, inactive = neutral standard material. Tapping selects the fold and the globe recolors live.

The “Mean nearest angle” score is the mean smallest angle per test camera to the nearest training camera (internally computed in radians, displayed in degrees in the UI).

IN PLAIN WORDS

One small card per fold on the right with the number of cameras and the average distance to the nearest training camera. Clicking it selects this fold as test.

BayesOpt Console (W30–W39)

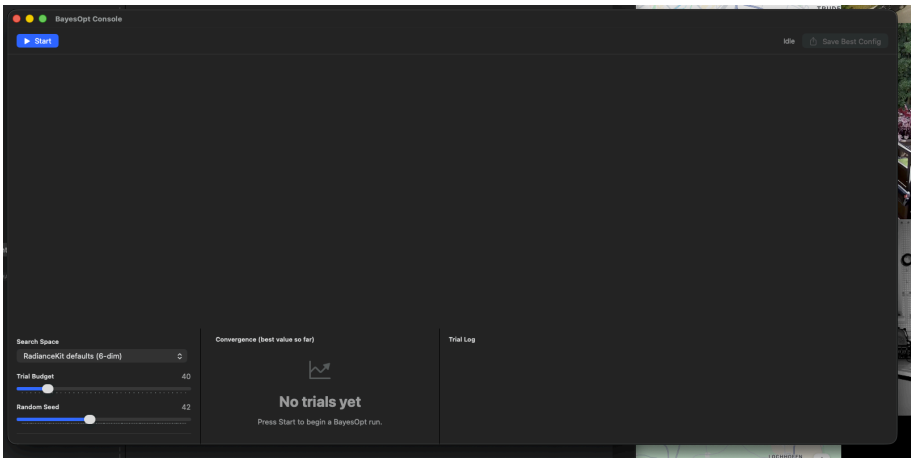

Figure 24: BayesOpt console — empty state before trial start

Empty state with search-space picker (`RadianceKit defaults (6-dim)`), trial budget slider (default 40), random seed (42) and three empty panels for convergence chart, trial log and search-space parameter list.

Empty state (after first open) — convergence chart and trial table fill up as soon as a run is started, see next shot.

Figure 25: BayesOpt console after 40 trials — convergence chart rises steeply until trial 15, best value 0.9943, trial log with init/bo/restart tags

WHAT'S IN THE IMAGE Status top right "Finished — best 0.9943 after 40 trials". Left sidebar: search-space picker on `RadianceKit defaults (6-dim)`, trial budget 40, random seed 42. Parameter list shows the six hyperparameters to tune with their value ranges: `mipSmoothing3DScale [0.05, 0.5]`, `mipFilter2DVariance [0.1, 0.6]`, `densifyGradThreshold [5e-07, 5e-06]`, `ssimWeight [0.05, 0.5]`, `mcmcNoiseScale [1e-05, 0.0001]`, `mcmcRelocationInterval [50, 200]`. Center: convergence chart (X = trial index 1–40, Y = objective value 0–1) — gray points = initial samples (LHS), blue points = BayesOpt acquisition, orange points = restart trials (#22 and #31). Best-value line rises steeply up to trial ~7, then only marginal improvement until trial 15, from there a flat plateau at 0.99+. Right sidebar: trial log #1–#34 with score + tag (init/bo/restart). Save Best Config button top right writes `bayesopt-best.json`.

What it is: A Bayesian-optimization console for hyperparameter search. Bayesian optimization is an automatic method that tries to find the optimum of an unknown function with as few experiments as possible — typically: "which combination of mcmcMaxGaussians, capMultiplier, ssimWeight and gradThreshold delivers the best PSNR for my scene class?" Instead of a grid of 6^4 = 1296 trials, Bayesian optimization tries about 40–100 informed trials and gets close to the optimum that way.

Important: The version currently shipped in the app does not run the optimization against real training runs (that would take days) but against a synthetic demo objective — a multi-modal landscape with hill-climbing character plus light noise. This is intentional: the window is meant to show you the behavior of the optimizer (convergence curve, sample points, best-so-far) and let you understand the search-space definitions. For real training-driven BayesOpt runs (as carried out in Phase Q7 for the scene-class presets), a separate offline CLI workflow is used; the window is the live UI variant.

WHEN TO OPEN Three use cases: 1. You want to understand how BayesOpt works — then start a demo run and observe the convergence chart. 2. You're planning a new scene class (such as "aquariums" or "antique furniture") for which the built-in 10 presets don't fit perfectly. Mentally define a search space, test it here with the "Bowl demo" or "Densify" preset, then export the best config as JSON and use it as a starting point for a real training run. 3. You want to inspect the default search spaces defined in the RK-

BayesOpt package (Mip subset, RadianceKit defaults) — they are listed in the parameter panel of the left sidebar.

HOW TO INTERPRET - **Convergence chart** (middle column): Y = best objective function value achieved so far. X = trial index. Initially rises steeply (BayesOpt tries the initial samples randomly, some of them lucky), then flattens out increasingly because the near-optimum region is exhausted. If the line stays flat for 20+ trials, you can stop the run — additional trials won't add anything. The individual points in the chart are the individual trial values (so not "best so far"), colored by phase: gray = initial sample, blue = BayesOpt acquisition, orange = restart. - **Trial table** (right column): #1, #2, #3, … each with value and phase tag. The best trial so far is marked with a yellow star. From the table you can identify the best trial and inspect its parameter values later during export. - **Search-space inspector** (left sidebar): shows for the selected preset all parameter names and their search ranges `[lo, hi]`. If you're on the preset "RadianceKit defaults (6-dim)", you see e.g. "densifyGradThreshold [5e-7, 5e-6]" — so log-uniform between these two values.

Pick preset "RadianceKit defaults (6-dim)", trial budget 40, seed 42. Click "Start". Observe: the first 8 trials are gray (initial samples, LHS Latin hypercube), the following ones blue (BayesOpt-acquired). The convergence chart rises steeply up to trial ~15, after that it flattens out. At trial ~30–40 the best value stabilizes. Click "Save Best Config" — a `bayesopt-best.json` is saved with the preset name, trial index, value, and the decoded parameter values. You can then manually copy this JSON into your preset definition.

W30 Button "Start"

WHERE

Toolbar on the left, in idle/finished state..

TECHNICAL

Resets the trial list, switches into running state, generates a new run ID (for stale detection on multiple Start clicks) and creates a fresh pause gate. Then a background task starts that runs the optimizer as an asynchronous stream. Initial-samples size is `min(8, budget / 4 + 1)` — so typically 8 Latin-hypercube samples at budget ≥ 28, fewer at small budget. Trial updates are received incrementally and appended to the list. Stale-run protection: if in the meantime a second Start click sets a new run ID, updates from the old run are discarded.

Primary action style for the prominent button look.

IN PLAIN WORDS

Starts a fresh optimization run with the current search space, budget and seed.

W31 Button “Pause”

WHERE

Toolbar on the left, in running state..

TECHNICAL

Activates the pause gate and switches into paused state. The actual effect: the runner waits in a 50 ms polling loop before it evaluates the next objective function. This means a trial currently running is run to completion (it’s synthetic and only takes microseconds), but no further trial is started. As soon as Resume runs, it continues where it left off.

IN PLAIN WORDS

Pauses the run. The current computation still runs to completion, then it pauses.

W32 Button “Stop”

WHERE

Toolbar on the left, in running and paused state..

TECHNICAL

Cancels the runner task, nulls the reference, releases the pause gate (if still paused) and switches into finished state (if trials exist) or idle state (if not). The already computed trials remain visible in the list — Stop does not delete them. Destructive button role shows the button in red because it cancels the run.

IN PLAIN WORDS

Cancels the run permanently. Trials stay visible; you can still export the best config.

W33 Button “Resume”

WHERE

Toolbar on the left, in paused state..

TECHNICAL

Releases the pause gate and switches back to running state. The runner task is already running (it’s waiting in the polling loop); as soon as the loop notices that the pause has been lifted, it continues and starts the next trial.

IN PLAIN WORDS

Resumes a paused run.

W34 Button “Save Best Config”

WHERE

Toolbar on the right, always visible (but disabled if no bestTrial exists)..

TECHNICAL

Opens a save dialog with default filename `bayesopt-best.json`, restricted to JSON. After confirmation a payload dictionary is built: preset name, trial index, value (objective score), parameters (dictionary of decoded parameter names → values). The decoding projects the normalized search-space coordinates in [0,1]^d back into the original value range (with log-uniform/linear/integer scales accordingly). JSON output is pretty-printed and with sorted keys. On write errors (in the current demo version) is silently ignored — no error UI because it’s a demo path.

The button stays gray as long as no trial has run.

IN PLAIN WORDS

Saves the parameter values of the best trial so far as JSON. You can then manually copy these values into your preset configuration.

W35 Picker “Search Space” preset

WHERE

Left sidebar, at the top..

TECHNICAL

Menu picker with four preset options: - “RadianceKit defaults (6-dim)” — the full standard search space with all Q7 hyperparameters. - “Mip subset (2-dim)” — only `mipSmoothing3DScale` [0.05, 0.5] log-uniform and `mipFilter2DVariance` [0.1, 0.6] linear. Useful when you want to tune Mip-Splatting for a scene class. - “densify-until + ssim-weight + grad-thresh” — three Densify-relevant parameters (`densifyGradThreshold` log-uniform, `ssimWeight` linear, `densifyUntilIter` integer). - “Bowl demo (1-dim)” — pedagogical single-parameter search space for “this is how BayesOpt works” demos.

While a run is active, the search space cannot be switched (would confuse the optimizer).

IN PLAIN WORDS

Which hyperparameter search space BayesOpt explores. Default is “RadianceKit defaults”. For targeted Mip-tuning attempts “Mip subset”. To understand how BayesOpt works “Bowl demo”.

W36 Slider “Trial Budget”

WHERE

Left sidebar, below the search-space picker..

TECHNICAL

Slider from 10 to 200, step size 5. Default 40. This means: BayesOpt may do a maximum of N trials. Of these the first few are initial samples (Latin hypercube), the rest are real BayesOpt trials. Rules of thumb for practice: a search space with d dimensions needs about 10*d to 20*d trials for a good optimum. At 6-dim defaults that’s 60–120, at 2-dim Mip subset 20–40, at 1-dim Bowl demo 10–20.

During the run the slider is disabled.

IN PLAIN WORDS

How many optimization attempts at most. More attempts = better solution, but costs more time. 40 is a good default for the demo objective.

W37 Slider “Random Seed”

WHERE

Left sidebar, below the budget slider..

TECHNICAL

Slider from 1 to 100, step size 1. Default 42. The seed is passed both to the initial Latin-hypercube samples and to the noise component of the demo objective. Reproducibility: same seed + same search space + same budget yields exactly the same trial sequence. Useful for “do all your colleagues get the same run when they rebuild the demo?”. Disabled during the run.

IN PLAIN WORDS

Controls the random generator. Same seed = same run — for re-producing.

W38 Chart (Convergence)

WHERE

Middle column of the window..

TECHNICAL

Swift Charts diagram with two layers: 1. a line for “best-value-so-far” per trial — a monotonically rising or constant curve in accent color. 2. one point per trial with the individual objective value, colored by phase. Symbol size 40. Three phase labels: “init” (gray), “bo” (blue), “restart” (orange).

A small legend shows the phase colors at the top left. If the trial list is empty (before the first start), an empty-state display with a chart icon and the hint “Press Start to begin a BayesOpt run.” is displayed instead.

IN PLAIN WORDS

The progress chart. The solid line is “best solution found so far”; the points are the individual attempts. If the line stays flat for a long time, BayesOpt has found the optimum.

W39 Table (Trial Log)

WHERE

Right column of the window..

TECHNICAL

Scroll area with lazily stacked trial rows. One horizontal stack per row: trial number (3-digit monospaced, on the left), value (monospaced, right-aligned, 70 pt wide), phase tag (capsule, filled with phase color at 25% opacity), optionally a yellow star if this trial is currently the best. An auto-scroll mechanism automatically jumps to the end as soon as a new trial is added — so you can follow the live progression at the bottom of the screen without scrolling yourself.

IN PLAIN WORDS

The table of all attempts. Value, phase, star for the best. Autoscrolls along, new trials appear at the bottom.

Main Window: Loss Curve and Gaussian Count (I39–I41, cross-reference)

Three of the Inspector displays in the main window deserve their own explanation because they are constantly visible during a running training and there are important rules of thumb for when the curve looks healthy. The displays are in the Inspector under the “Loss Chart” section (see Chapter 2 — Inspector) and complement the Holdout analysis from the auxiliary window above.

When is the loss curve healthy? A healthy loss curve shows three phases: (1) **Warmup** — the first 200–500 iterations the loss falls steeply from high (typically 0.15–0.25 for L1+SSIM combined depending on the scene) to about half. If the loss does NOT fall in this phase, the input is usually wrong (broken images, bad SfM poses, too few initial Gaussians). (2) **Densification** — between ~500 and densifyUntilIteration (classic 15K, MCMC up to 20K or 25K) the loss continues to fall, often with small jumps downward when densify operations insert new Gaussians and the optimizer exploits them. The Gaussian count rises in this phase. (3) **Refinement** — after that the loss runs into a tail that flattens out. Typical end values: Tanks-&-Temples Truck with P4 Quality lands at L1 ≈ 0.023, Horse with Full Classic V546 at L1 ≈ 0.0230, outdoor Mip-NeRF360 scenes often worse (0.04–0.07).

What does a plateau mean? A plateau (loss curve runs horizontally over several thousand iterations) has two interpretations: (a) the model has converged, more training won't add anything — the good case. (b) the model is stuck (local minimum, bad gradient information, a cap at the buffer limit) — the bad case. Both look identical in the chart. Distinction: look at the Gaussian count. If it's also flat AND close to the MCMC cap (e.g. 150K of 150K at `.fullMCMC`), you are at the limit — either raise the cap or accept the plateau. If the Gaussian count is still growing but the loss isn't falling, it's stuck.

When to abort vs continue training? Rule of thumb: 10K iterations long no improvement of the min loss → abort, further iterations are wasted. Before that: you can append an extension via Cmd+T (Training menu → Continue Training → +5K iterations), if you see marginal improvement. Watch out: with MCMC the plateau is often real — the cap is the natural limit.

Gaussian count plateau is NOT a "done" signal. It only means that MCMC has reached the cap or that Classic Densification is exhausted. The real "done" question is answered only by the Holdout analysis — PSNR/SSIM/LPIPS on an independent test set, evaluated in the Holdout window (W23–W29) or via the `--benchmark` flag.

PSNR/Holdout is the truth, loss is only a proxy. Loss is a relative metric: it falls as your model fits the training views. A low loss does not automatically mean a good model — if the model has memorized the training images (overfitting), the loss would be small, but PSNR on unseen views (Holdout) would be bad. Therefore: for final quality assessment always look at Holdout metrics, not end-loss alone.

Rule-of-Thumb Box

- User Guide and Keyboard Shortcuts are static help — for keyword questions fast, for depth use this manual at hand.
- Open Manage Storage as soon as the disk falls below 10% free space. Logs and imports staging are the usual culprits.
- Pareto Dashboard only useful after at least three or four training reports. X-axis = costs (Time / Gs), Y-axis = quality (PSNR / SSIM). The Pareto front shows the efficient combinations.

- Use Holdout Analysis before publishing PSNR benchmarks with others — it assures you that your test set is really representative.
- BayesOpt Console is primarily a learning and inspection tool for search-space definitions. For real training-driven hyperparameter tuning use the offline CLI workflow.
- Loss plateau and Gaussian-count plateau are to be interpreted separately. Cap limit is not a “done” signal. Real quality is measured only by Holdout PSNR.
- 10K iterations without min-loss improvement → stop training.

CHAPTER

Chapter 6 — Training Configuration

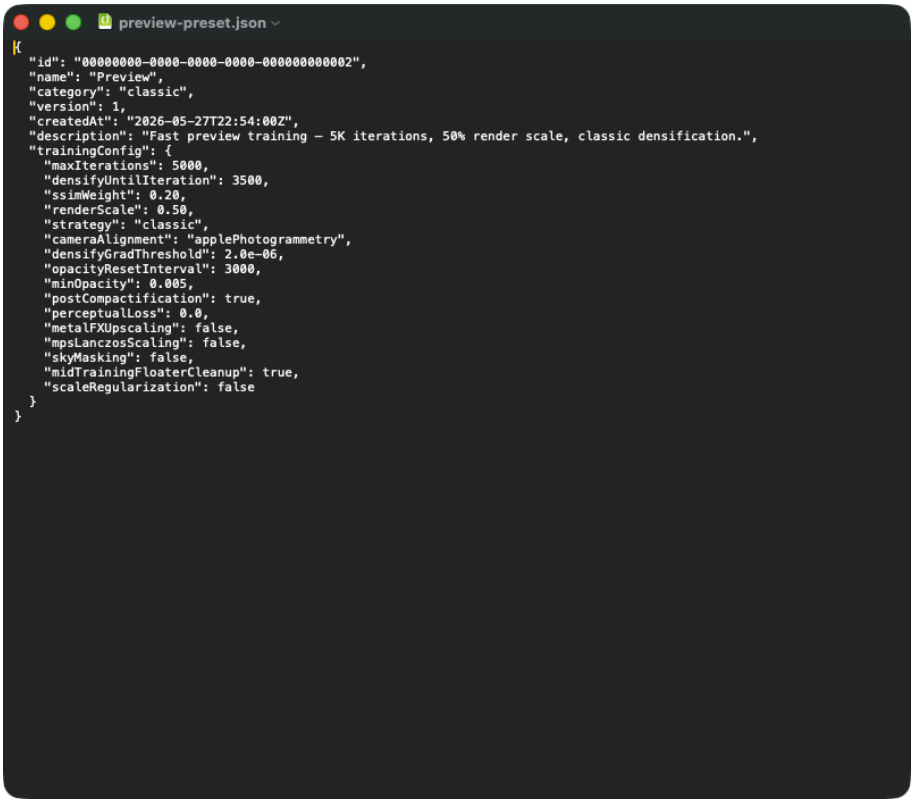

Figure 26: Preview preset exported as JSON and displayed in TextEdit — fields id/name/category/version/createdAt/description, trainingConfig with all relevant parameters (maxIterations 5000, densifyUntillIteration 3500, ssimWeight 0.20, renderScale 0.50, strategy classic, cameraAlignment applePhotogrammetry, densifyGradThreshold 2.0e-06, opacityResetInterval 3000, minOpacity 0.005, six boolean toggles)

WHAT YOU SEE IN THE IMAGE A typical preset JSON export. Top-level fields: `id` (UUID), `name`, (classic | mcmc | sceneClass | custom), (schema version), (timestamp), (free text). The nested object holds the parameters that are critical for reproducibility — on import the entire block is deserialized into the `TrainingConfig` struct, and defaults from the current app version fill in any fields missing from the JSON (e.g. after an app update). To hand a preset over to another Mac, you just ship this JSON file.

The `TrainingConfig` struct is the heart of every training run in RadianceKit. It collects every parameter that influences training — from maximum iteration count over the eight learning rates to the special fields for MCMC, Mip-Splatting, the curriculum and the scene-aware cap logic. You edit it in the sidebar in the Training Configuration section (Expert View), save it as a preset or hand it over as a JSON export to another Mac. At training start this very object is frozen and handed to the GPU backend.

This chapter is reference material for power users and script authors. It lists all 81 public fields, the 9 static presets and the one public method. The source file is `TrainingConfig.swift` — when in doubt the doc comment stored there and the initializer default are the source of truth.

> NOTE · UI VS. PRESET/CLI
>
> Only 12 of the 81 fields have a direct slider, toggle or picker in the Inspector (sandboxed App Store build): **T1, T2, T17, T20, T22, T38, T56–T58, T60, T61, T73**. The remaining 69 fields are set via the chosen **preset** (Chapter 7) and can only be overridden directly via a **CLI flag** (see Chapter 5). This separation is by design: defaults stay stable and production-tested, while power users still have an escape hatch. If a field really interests you, first check Chapter 2 (Inspector) and Chapter 5 (CLI) to see whether you can reach it without JSON tinkering.

Table of contents:

Iteration (T1–T2)

T1 maxIterations

DETAILS

Default: 30 000 (initializer), 35 000 (`.full`), 200 000 (`.fullMCMC`) **Range:** 1 000 – 500 000 (UI slider), no hard upper limit in the logic **Defined in:**

TECHNICAL

Total number of training iterations the backend runs through. One iteration means a forward render of a single training camera, one backward pass over all loss components (L1 + SSIM + optional regularizations + sky mask) and one Adam optimizer step. This number directly drives the other schedules: position learning rate follows a cosine annealing curve from 0 to either `T1` itself or to `T49 positionLRScheduleEndIteration`; densification stops at `T2 densifyUntilIteration`; MCMC noise decay ends at `T69 mcmcNoiseDecayEnd`; SH degree upgrades happen at the three marks in `T21` .
For classic densification the empirically determined sweet spot is at 20 000–35 000 iterations (Sessions 1–32, V546 tests), for MCMC at 60 000–200 000 (V534). Pushing well beyond the values stored in the preset rarely brings additional quality — Adam momentum saturates, and without an LR decay end the loss stagnates. Conversely, going below ~5 000 leads to incompletely converged geometry (density control has too little time to clone/split).

IN PLAIN WORDS

How long the app trains. More iterations = better result, but eventually no longer noticeably better, just much longer. The presets are chosen so that without thinking you get a good value: Quick 1 000, Preview 5 000, Balanced 20 000, Quality 35 000, MCMC Quality 200 000. If you turn the knob yourself: with MCMC feel free to go high (100 000–200 000), with Classic not above 40 000 — doesn't help any more after that.

T2 densifyUntilIteration

DETAILS

Default: 15 000 (initializer), 5 000 (`.full`), 160 000 (`.fullMCMC`) **Range:** 0 – **Defined in:**

TECHNICAL

Iteration at which densification stops. Up to this point Gaussians are cloned, split and pruned according to the rules parameterized in `T11–T16` (Classic) or `T67–T70` (MCMC); after that the Gaussian count stays constant and only positions, rotations, scales, opacities and SH coefficients are optimized (refinement phase). In the 3DGS original paper the value sits at 50 % of `T1` , in RadianceKit's `.full` preset at only ~14 % (5 000 of 35 000) — a consequence of the V310/V338 experiments which showed that after 5 000 iterations further densification makes the result worse (more floaters, higher memory use, no quality gain). MCMC, on the other hand, runs relocation up to 80 % of `T1` (V504b) because MCMC does not produce harmful floaters. If `T2` is chosen too small (< 1 000), too few Gaussians arise; too large under Classic (> 50 % of `T1`) leads to overgrowth and RGB saturation outliers (see Outdoor Overtraining Findings).

IN PLAIN WORDS

Up to when the app is allowed to create new Gaussians. After that, only what already exists is refined. With classic training at 35 000 iterations, 5 000 is the right value here — anything above makes the scene mushier. With MCMC it is 80 % of total iterations (so 160 000 for a 200 000 run). If you change the Quality preset, better leave this field alone.

Learning Rates (T3–T10)

T3 positionLearningRate

DETAILS

Default: 0.00016 **Range:** 1e-7 – 1e-3 (recommended) **Defined in:**

TECHNICAL

Adam learning rate for each Gaussian's XYZ position at the start of training (iteration 0). It follows a cosine annealing curve and decays over training down to `T4 positionLearningRateFinal`. The default 0.00016 comes from the 3DGS original paper (Kerbl et al. 2023) and in RadianceKit is not to be scaled even with higher image resolution — position moves in world coordinates, not in pixel space. A clear increase (> 0.0005) makes Gaussians jump over long distances and the loss becomes unstable; values well below (< 0.00005) mean badly initialized point clouds never find their place. V414 tested doubling the initial value → 16.8 % worse L1 loss; the V544a tunings confirmed the paper default as optimal. Note: under `.fullMCMC` we deliberately leave this at the default — MCMC needs constant learning rates for its relocation logic, so tuning here brings nothing.

IN PLAIN WORDS

How fast the splat points are allowed to move through space. The default value is very well tuned and basically doesn't need changing. Only when splats "wobble" in the image or a whole corner is missing because nothing moves there would the learning rate be something to adjust — but then typically something else is wrong upstream (camera poses, initial point cloud).

T4 positionLearningRateFinal

DETAILS

Default: 0.0000016 (initializer + paper), 0.000016 (`.full` , `.fullMCMC` — 10× higher) **Range:** 0 – **Defined in:**

TECHNICAL

End value of the position LR cosine annealing curve. It is reached either at `T1 maxIterations` or, if set, at `T49 positionLRScheduleEndIteration`. The RadianceKit `.full` preset uses 0.000016 — i.e. 10× higher than the paper default 0.0000016. V420 experiments showed that 0.5× of the final value (0.000008) makes loss 6.4 % worse; V414 showed that 2× the initial value makes it 16.8 % worse. The high final value is not a trade-off but a deliberate choice: with too aggressive a decay, Gaussians lose during the refinement phase the ability to react to newly arrived densification candidates. Through the V431/V433 extension the schedule phase can be shortened (`T49 < T1`), so that `T4` is reached before the end of training and the rest of training runs at the constant mini-LR — typical configuration: `T49 = 20 000` , `T1 = 35 000` , refinement thus at 0.000016 for 15 000 iterations.

IN PLAIN WORDS

How slow the position learning rate becomes at the end of training. We've intentionally set this less aggressively than the original paper — splats can still wiggle a bit until the very end, which makes them sharper. If you turn the knob: higher = more restless splats at the end, lower = splats can no longer adapt when new ones appear.

T5 shDCLearningRate

DETAILS

Default: 0.0025 (initializer + paper), 0.005 (`.full` and all MCMC presets — 2×) **Range:** 0.0001 – 0.05
Defined in:

TECHNICAL

Adam learning rate for the DC component (degree 0, i.e. constant albedo) of the spherical harmonic color. SH-DC corresponds to the direction-independent base tone of a Gaussian, essentially the "base color". V176 and V188 experiments found 2× higher than the paper default to be optimal — faster color convergence, especially because with short training (, 5 000 iterations) the SH-DC otherwise does not converge. Unlike the geometric LRs, SH-DC has no decay; the learning rate stays constant over all iterations (or just follows the optional extended-phase decay from `T51`). V416 tested a quadrupling to 0.01 → 6.4 % worse loss with beta2=0.99 Adam.

IN PLAIN WORDS

How fast the base color of each splat adapts. You almost never change this value yourself — the presets have the right value. Higher would be faster but can lead to unstable colors.

T6 shRestLearningRate

DETAILS

Default: 0.000125 (initializer + paper), 0.00025 (`.full` and MCMC — 2×) **Range:** 0.000001 – 0.005 **Defined in:**

TECHNICAL

Adam learning rate for the higher-order SH coefficients (degree 1, 2, 3 — i.e. the view-direction-dependent color components that produce highlights, reflections and soft shading). 20× smaller than `T5` per paper convention, because these coefficients grow quadratically in count (3 for degree 1, 5 for degree 2, 7 for degree 3 → 15 floats total per Gaussian) and without a smaller learning rate would oversaturate the image. They are unlocked in two steps — until the first mark in `T21 shDegreeUpgradeIterations` only degree 0 is active (so only `T5`), after that 1, then 2, finally 3. Low values here are particularly important on scenes with lots of diffuse lighting; on very glossy surfaces (car paint, water) tuning doesn't help — the SH representation itself is limited.

IN PLAIN WORDS

How fast the view-dependent color effects (reflections, gloss) learn. By default very small, otherwise everything starts to shine. Better leave the value alone — if you want better highlights, you're better served by MCMC and longer training time than by this LR.

T7 opacityLearningRate

DETAILS

Default: 0.05 (initializer + paper), 0.1 (`.full` , MCMC — 2×) **Range:** 0.001 – 1.0 **Defined in:**

TECHNICAL

Adam learning rate for the logit opacity of each Gaussian. The app stores opacity as an unbounded float value and transforms it with sigmoid into [0, 1]; the LR acts in logit space. The paper default 0.05 was restored after V50 tests (best single-run L1 0.1664), V71 reverted V67's 0.025. The V188 doubling to 0.1 makes pruning more efficient — dead Gaussians fall faster below the `T14 pruneOpacityThreshold`. V418 showed: 0.05 with beta2=0.99 Adam is 7.1 % worse than 0.1 — the interaction with the Adam configuration is non-trivial. Low values (< 0.01) mean "dead" Gaussians linger forever and waste memory; too-high values (> 0.5) can lead to opacity explosion, which is why the logit value is clamped in the optimizer to [−15, 3] (see the "Opacity Explosion Prevention" note in CLAUDE.md).

IN PLAIN WORDS

How fast splats become transparent or opaque. Important for cleanup — splats that contribute nothing have to vanish quickly so no haze appears. The default value fits, only pros change it.

T8 opacityLearningRateFinal

DETAILS

Default: 0.0 (= “no decay”) **Range:** 0 or 0.001 – **Defined in:**

TECHNICAL

Optional cosine decay end value for the opacity LR (V427). When 0.0, decay is disabled and the opacity LR stays constant at `T7` over the entire training. V427 tested a decay 0.1 → 0.01 — result was 11.5 % worse loss; reverted, hence the default “off”. The hypothesis behind the field: in the refinement phase, a constant opacity LR could lead to oscillation, so that splats which had already reached the right level of transparency would be pushed around again by random gradient fluctuations. Empirically this is not the case — the logit clamping logic catches that anyway. The field stays available for future experiments; very long MCMC runs (> 500K iterations) might also benefit from it.

IN PLAIN WORDS

Whether the opacity learning rate should become smaller toward the end. Default: no. We tried it, it was worse, we leave it off. Stay at 0.

T9 scaleLearningRate

DETAILS

Default: 0.005 (initializer + paper), 0.01 (`.full`, MCMC — 2×) **Range:** 0.0001 – 0.1 **Defined in:**

TECHNICAL

Adam learning rate for the three scale components of each Gaussian in log space (RadianceKit stores log(scale) so that scales stay positive). The paper default 0.005, doubled in RadianceKit to 0.01 for better scale convergence with the optimized learning rate configurations. V423 experiment: 0.005 with beta2=0.99 Adam → 18.7 % worse loss and visibly too few Gaussians (density control couldn’t clone because scale updates were too sluggish). Scale controls the extent of each Gaussian — too fast learning leads to “needle” Gaussians (extremely long thin splats, see `T34 scaleRatioPruneThreshold`), too slow learning keeps splats too compact and density control has to split too often.

IN PLAIN WORDS

How fast the shape of the splats adapts. Default is good. If you push this up you get “needle” splats — extremely long thin droplets that make the image float.

T10 rotationLearningRate

DETAILS

Default: 0.001 (initializer + paper), 0.002 (`.full`, MCMC — 2×) **Range:** 0.0001 – 0.05 **Defined in:**

TECHNICAL

Adam learning rate for the four quaternion components of each Gaussian. The quaternion is re-normalized after each Adam update (L2 norm = 1) — otherwise the covariance matrix would degenerate. RadianceKit doubles the paper default in the Quality presets because rotation has smaller absolute gradient magnitudes than scale/position (on the unit sphere each step stays short) and without 2× rotation would be clearly under-converged in the 35 000 iteration window. V188 documents this. On NeRF-Blender scenes (Lego, Chair) rotation matters particularly — the edges of objects only line up properly after 5 000–10 000 iterations.

IN PLAIN WORDS

How fast the splats learn to rotate — that is, to orient themselves correctly on the surface of an object. Default fits. Put another way: if splats look like slabs lying askew instead of hugging the surface, training time is too short rather than this learning rate too low.

Densification — Classic (T11–T16)

T11 densifyGradThreshold

DETAILS

Default: 0.000002 (initializer, calibrated for 0.5× resolution), 0.0000011 (`.full` , calibrated for 1.0×), 0.000004 (`.quickTest` , calibrated for 0.25×), 2e-7 (`.fullClassicPaper`) **Range:** 1e-8 – 1e-3 (resolution-dependent) **Defined in:**

TECHNICAL

Threshold for the L2 norm of the screen-space projected gradient `dMean2D` , above which a Gaussian is marked for cloning or splitting. The absolute value depends directly on the training resolution — `dMean2D` scales roughly as 1/resolution² (more pixels = smaller per-pixel gradients). Hence every `T22 trainingRenderScale` step needs a calibrated threshold: 0.25× → 4e-6, 0.5× → 2e-6, 1.0× → 5e-8 … 1.1e-6 (`.full`). The paper default 0.0002 is NDC-normalized and not directly comparable to RadianceKit's world-space pipeline. With the V440-introduced `T52 adaptiveDensifyThreshold` flag the value can be derived at runtime from the p98 of the current gradient distribution — but V440 tested this on real scenes and produced 63 K Gaussians (catastrophic pruning loss); the flag stays off. Q5 (T77–T79) provides an alternative adaptive logic via rolling median. **This field is not harmless** — halving creates 2–4× more Gaussians (memory pressure, OOM risk); doubling can under-densify the scene.

IN PLAIN WORDS

How sensitive the app is when deciding whether a splat is under-represented and should be duplicated. Lower value = more sensitive = more splats. Higher = fewer splats. This is one of the most dangerous values of all: too low and your Mac fills up with millions of splats and may crash. Leave the field alone, or change it only in steps of 10 %.

T12 densifyFromIteration

DETAILS

Default: 500 **Range:** 100 – 5 000 **Defined in:**

TECHNICAL

First iteration at which densification becomes active. Before that only “bare” learning on the initial SfM point cloud happens, without new Gaussians being created. The default 500 comes from the 3DGS paper and gives initialization time to stabilize — if densification starts as early as iteration 0, mispositioned SfM points are cloned many times before they even find their proper place. V349 tested 1000 → slightly worse loss; the default is optimal.

IN PLAIN WORDS

When the app first starts cloning splats. Before that it just learns the already existing points. 500 is the default — gives the app enough time to find its bearings first before multiplying.

T13 densifyInterval

DETAILS

Default: 100 (initializer, MCMC), 200 (`.full`)
Range: 50 – 1 000 **Defined in:**

TECHNICAL

How many iterations sit between two densification steps. Paper default 100 — every 100 iterations the list of densify candidates is evaluated, cloned/split, and at the same time the list of prune candidates (sigmoid(opacity) < `T14 pruneOpacityThreshold`) is removed. V112 tests found 200 to be optimal for `.full` — this relieves the GPU because fewer reorganization passes run, and gives each Gaussian more time to settle after a clone action. V417 tested 100 with beta2=0.99 → 5.8 % worse (957 K Gaussians, over-densification). Under MCMC the same field is interpreted as the relocation interval; see `T67 mcmcRelocationInterval` for the MCMC-specific logic.

IN PLAIN WORDS

How often the app looks for new splats. 100 = often, 200 = medium. Higher means: each splat has more time to settle before being multiplied again. That’s good. Reducing to 50 can keep the GPU busy permanently without it being noticeably better.

T14 pruneOpacityThreshold

DETAILS

Default: 0.005 (initializer, paper, MCMC), 0.001 (`.full`) **Range:** 0.0001 – 0.1 **Defined in:**

TECHNICAL

Sigmoid opacity threshold below which a Gaussian is deleted at the next densification step. Works together with `T7 opacityLearningRate` and the logit clamp logic in the optimizer. V393 lowered the default from 0.005 to 0.001 in `.full` — result: splats that only matter under exotic viewing angles are kept longer and contribute to SH detail. V394 tested 0.0001 → slightly worse (too little pruned, memory wasted). Important: density control MUST always prune, even if buffer capacity is already full due to other measures (see "Density Control Must Always Prune" in CLAUDE.md) — otherwise dead Gaussians accumulate and the count freezes.

IN PLAIN WORDS

When a splat counts as "transparent enough" to be deleted. 0.005 is the paper default, in Quality we have 0.001 — i.e. we give splats a chance for longer. That makes soft light and faint shadows easier to represent. Setting higher (above 0.01) makes the splat count drop quickly — can make sense with memory pressure but costs detail.

T15 opacityResetInterval

DETAILS

Default: 3 000 (initializer + paper), 100 000 (`.full` = effectively disabled), 200 000 (`.fullMCMC` = disabled) **Range:** 1 000 – 100 000+ **Defined in:**

TECHNICAL

How many iterations between resetting the opacity of all Gaussians to a low value (~0.01) — a measure from the 3DGS paper to reassess "frozen" splats. V194 showed that with RadianceKit's warmup

1. stochastic training setup + 2× learning rates, opacity reset costs

5.5 % quality and that logit clamping already covers the reset function. Hence in `.full` practically disabled (100 000 > 35 000, so never triggered). V421 tested reset every 3 000 with beta2=0.99 → 4.9 % worse; reverted. Under `.fullClassicPaper` (Q1.5-A, paper-true test) it is deliberately set back to 3 000 — that was one of the levers with which the paper-magnitude Gaussian budgets were to be reached.

IN PLAIN WORDS

How many iterations between the app resetting the visibility of all splats to "almost invisible" — a kind of reset button for opacity. With us disabled (value so high that it never fires) because other mechanisms make it unnecessary. Only enable for paper-faithful experiments.

T16 maxScreenSize

DETAILS

Default: 0.0 (= disabled) **Range:** 0 (off) or > 0 **Defined in:**

TECHNICAL

Maximum screen-space size (in projected pixels) a Gaussian may reach before being forcibly split. The value is set to 0 (V48 tested and reverted) — RadianceKit's density control instead uses the world-space scale threshold from the `dMean2D` logic. Stays in the field catalog because future experiments with Mip-Splatting (T74–T76) or scene-specific splatting strategies could benefit from it. Enabling (value > 0, e.g. 20) would force splats that have grown very large on screen to subdivide — relevant with large smooth wall surfaces where a single giant splat offers too little detail.

IN PLAIN WORDS

Limit on how large a single splat is allowed to get on screen. With us off. Turning it on would cause huge flat splats (e.g. on a wall) to be forcibly broken up into several smaller ones. Leave off unless explicitly experimenting with it.

Loss (T17–T20)

T17 ssimWeight

DETAILS

Default: 0.2 (initializer + paper + `.full`), 0.05 (all MCMC presets) **Range:** 0.0 – 1.0 **Defined in:**

TECHNICAL

Weight of the D-SSIM term in the combined loss function `loss = (1 − λ) ∗ L1 + λ ∗ D-SSIM`, where λ = `T17`. The 3DGS paper default 0.2 is optimal for classic densification — V383 tested 0.3 → 28.9 % worse, V373b confirmed 0.2 as the sweet spot. For MCMC it was independently established in V521b/V534: 0.05 is optimal because MCMC, through its stochastic exploration, needs a stronger L1 signal component — higher SSIM weights would dilute the relocation decisions. SSIM is significantly more expensive to compute than L1 (local 11×11 windows over the whole image); RadianceKit uses an MPS-accelerated implementation that stays under 1 ms per 1080p image. Q7 BayesOpt sweeps found scene-specific optima between 0.05 (`.outdoorPreset` : 0.082) and 0.171 (`.indoorPreset`).

IN PLAIN WORDS

How important the app considers "structures are similar" alongside "every pixel matches". 0.2 is the default and gives a good image. Lower = more pixel-accurate but can result in softer transitions. Higher = more structurally similar but details get softer. Let the presets decide.

T18 ssimWeightRefinement

DETAILS

Default: 0.0 (= “no switch, keep ssimWeight”)
Range: 0 or 0 – 1.0 **Defined in:**

TECHNICAL

Optional SSIM value for the refinement phase after `T2 densifyUntilIteration`. V428 tested 0.2 → 0.3 in refinement → 16 % worse loss (both L1 and SSIM degraded); reverted, hence default 0.0. The hypothesis behind the field was that after densification — when no more new Gaussians are being created — a stronger SSIM share would maximize structural sharpness. Empirically false: increasing SSIM weight means indirectly lowering L1 weight, and L1 is the much more meaningful signal in the final refinement phase. The field stays available for future experiments with perceptual loss (T60) or edge loss (T19), where a refinement-specific loss composition might make sense.

IN PLAIN WORDS

Special setting for the second training phase (refinement after splat duplication). At 0.0: same SSIM weighting as before. Tweaking brings nothing empirically, so off.

T19 edgeLossWeight

DETAILS

Default: 0.0 (= disabled) **Range:** 0 or 0.001 – 1.0
Defined in:

TECHNICAL

V437 experimental loss: weight of a Sobel-gradient-domain L1 loss that compares image edges directly (ground-truth Sobel vs render Sobel) on top of L1+SSIM. Hypothesis: edge information is a perceptual cornerstone of image quality and an explicit term should encourage Gaussians to hit edges better. Test results: weight 0.1 → 11 % worse loss, 0.01 → quality-neutral but 10 % slower. The Sobel pass costs an extra MPS forward on ground truth and render. Hence permanently disabled. Future use case: scenes with hard artificial edges (architecture, furniture, renderings) could benefit — Q7 scene class presets didn’t pick it though, instead they scaled the SSIM weight.

IN PLAIN WORDS

Experimental extra that treats edges as particularly important. Brings nothing empirically. Stays off.

T20 skyMaskingEnabled

DETAILS

Default: false (initializer and all presets) **Range:** boolean **Defined in:**

TECHNICAL

Enables sky masking. In every image the sky region is masked out via Apple Vision Framework (`VNGenerateForegroundInstanceMaskRequest`), and the loss in this region is set to zero. Reason: outdoor scenes often suffer from blue/gray/white sky pixels driving the app to place Gaussians exactly there — which is perceived as “floaters”. Without a sky mask the loss in this region would never be zero because the sky in the image varies slightly and the app keeps trying to rebuild it with splats. The Vision mask is computed once per camera before training and held in RAM. Typically activated together with `T45 skyDomeEnabled` (UI logic in the Settings view). Leave disabled for indoor scenes or synthetic renderings — the mask would erroneously detect ceilings or walls as “sky”.

IN PLAIN WORDS

Enables a special mode for outdoor captures: the sky is ignored during training, so the app doesn’t try to rebuild it with splats. Recommended for every outdoor scene. Leave off for indoor or for 3D renderings from Blender.

SH Degree Progression (T21)

T21 shDegreeUpgradeIterations

DETAILS

Default: `[1_000, 2_000, 3_000]` (initializer), `[2_000, 5_000, 8_000]` (`.full` , MCMC), `[1_000, 2_000]` (`.preview` — degree 3 skipped) **Range:** `[Int]` , each value in `[0, maxIterations]` , monotonically increasing **Defined in:**

TECHNICAL

Iterations at which the active SH degree is upgraded 0→1, 1→2, 2→3. Before the first mark only the DC components are active (i.e. `T5 shDCLearningRate`), after the first mark DC + 3 degree-1 coefficients, after the second mark + 5 degree-2 coefficients, after the third mark all 15 coefficients. Memory consumption per Gaussian grows in steps accordingly — 4 floats → 16 floats → 36 floats → 64 floats. The Quality presets delay the upgrades compared to initializer defaults (V228) because the geometry should stabilize first, before the color details with their higher frequency are added on top. V384 tested `[1K, 2K, 3K]` for `.full` → 9.3 % worse — confirms the delay. `.preview` caps at degree 2, because degree 3 doesn't converge in 5 000 iterations and just consumes optimizer capacity. Q6 (T80–T81) offers an alternative curriculum logic that dynamically overrides this list.

IN PLAIN WORDS

At which points during training the app learns that colors can look different from different viewing angles (highlights, reflections). Only late — so first the shape is right, then the color. The values in the presets are set so this works well. Don't change anything unless you know exactly why.

Performance (T22–T25)

T22 trainingRenderScale

DETAILS

Default: 1.0 (initializer, `.full`, MCMC, SceneClass), 0.5 (`.preview`), 0.25 (`.quickTest`) **Range:** 0.05 – 2.0 (typically 0.25, 0.5, 1.0) **Defined in:**

TECHNICAL

Rendering resolution during training relative to the original resolution of the training images. At 0.5 every image is downscaled to 50 % width × 50 % height (i.e. 25 % of the pixels) and the Gaussian rendering happens at this lower resolution. Reduces both memory and compute quadratically. Important: `T11 densifyGradThreshold` has to match the chosen resolution — the gradient magnitudes scale with 1/resolution², which is why `.quickTest` (0.25×) has a much higher threshold (4e-6) than `.full` (1.0×, 1.1e-6). RadianceKit warns at very large images and adjusts automatically — 3 MP target resolution. With extreme 4K input images, 0.5 or even 0.25 would make sense, otherwise any Mac will run only in CPU compaction.

IN PLAIN WORDS

How big the images are during training. 1.0 = original, 0.5 = half size. Half size = four times faster but the finest details are missing. The presets pick the right value; with extremely large input images (over 12 megapixels) the app switches down automatically.

T23 resolutionWarmupScale

DETAILS

Default: 0.0 (= disabled) **Range:** 0 or 0.1 – **Defined in:**

TECHNICAL

V133 optimization: train the densification phase (iter 0 to `T2`) at a lower resolution than the refinement phase. V308 turned it off again for `.full` because with `T22 = 1.0` and cosine annealing, the time win was marginal and quality suffered slightly. Stays in the field catalog because it could become useful again with 4K inputs and long training runs — Q6 curriculum (T80) picked up a similar logic, though there it is tied to the LR schedule. If enabled and `T80 curriculumResolutionRamp` is also true, Q6 wins and overrides this value.

IN PLAIN WORDS

Special feature: in the first training half learn from smaller images, in the second from large ones. Saves time. Off because the newer Q6 variant solves it better.

T24 tileSize

DETAILS

Default: 16 **Range:** 8, 16, 32 **Defined in:**

TECHNICAL

Size of the rasterization tiles in pixels. The Gaussian Splatting rendering is tile-based: the image is divided into 16×16 pixel tiles, each tile collects the Gaussians relevant to it, sorts them by depth and blends them. 16 is the standard used by practically all 3DGS implementations and is hard-coded in RadianceKit's Metal kernels; changing this value would require shader recompilation and is not effective in the current state. Stays as a field in case a future engine version supports dynamic tile size.

IN PLAIN WORDS

Internal render parameter. Default 16, do not change.

T25 throttleDelayMs

DETAILS

Default: 0 (initializer, `.full`, MCMC, SceneClass), 0 (`.preview`) **Range:** 0 – 100 **Defined in:**

TECHNICAL

Artificial delay between training iterations in milliseconds. 0 = full speed (default). Higher values make the Mac more "usable" during training, by giving GPU/CPU regular breathers — the responsiveness of other apps improves, but training time grows linearly with the delay. Typical values: 1–2 ms ("light" throttling, +5 % training time, Mac feels more responsive), 5 ms ("medium", +15 % training time), 10+ ms ("eco", potentially double the training time). Offered in the Inspector under "Performance" but not in the default view — see backlog `dev_ux-backlog.md` which suggests removing it from the Expert View because misunderstood it dramatically extends training time.

IN PLAIN WORDS

How many milliseconds of pause the app makes between training steps. 0 = no pause, as fast as possible. Higher values make the Mac more usable during training — but the training takes longer too. On an M3 Ultra or Mac Studio you can leave this at 0; on a MacBook Air 2 or 5 would be a good value.

Diagnostics and Point Cloud Preparation (T26–T30)

T26 depthDistortionWeight

DETAILS

Default: 0.0 (= disabled) **Range:** 0 or 0.0001 – 0.05
Defined in:

TECHNICAL

V366 experimental: weight of a depth distortion regularization loss. Penalizes Gaussians that, along a render ray, are stacked in depth but conceptually belong to the same surface — this encourages concentrated depth distributions and reduces floaters. Tests: 0.01 → 4.5 % worse, 0.001 → 8.1 % worse. The theoretical advantage — improving multi-view consistency — does not show up in the L1 loss, because the hypothesis implicitly assumes that the SfM geometry is correct and the Gaussians just need to be "stacked". In practice the SfM point cloud is usually the weakest component, not the stacking. Stays available for multi-view datasets with particularly clean poses (synthetic, Mip-NeRF 360 with ground truth).

IN PLAIN WORDS

Experimental feature to avoid multiple splats lined up at the same spot. Not enabled because the tests showed no benefit.

T27 singleViewOverfit

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNICAL

Diagnostic flag: when true, every training iteration must use camera index 0 instead of a random one from the camera pool. Reason: if the model can't even overfit a single view (i.e. the loss on view 0 doesn't go to zero even after 10 000 iterations), there's a fundamental bug in the forward/backward pass. This switch was used intensively during the development of the Metal shaders and the differentiable rasterizer kernels — V42–V47 phase. Today only available as a sanity check if someone has modified backend code and wants to do a regression test. Via CLI with `--single-view`.

IN PLAIN WORDS

Test mode for developers. They can use it to check whether the app can even learn from a SINGLE image. Irrelevant for normal users, always leave off.

T28 maxCameras

DETAILS

Default: 0 (= “use all cameras”) **Range:** 0 or 1 – N **Defined in:**

TECHNICAL

Diagnostic limit from V43: train only with the first N cameras, ignore all further ones. Original reason: test the hypothesis that too many cameras create gradient conflicts (too many conflicting loss signals for the same Gaussian). Test result: no systematic advantage from artificial limiting — more frames practically always bring more quality. Stays as a CLI flag (`--max-cameras N`) for targeted experiments, e.g. “does training work on the first 100 images of a 1 500-image drone flight?” Not exposed in the UI.

IN PLAIN WORDS

Diagnostic field for developers — use only the first N images, ignore the rest. Normal user doesn’t need this, value at 0 = all images. More images = better result (see `feedback_more-frames-better.md`).

T29 maxInitialPoints

DETAILS

Default: 0 (= “use all SfM points”) **Range:** 0 or 1 000 – 200 000+ **Defined in:**

TECHNICAL

V54 safety net: limits the number of initial SfM points the training starts with. Dense COLMAP reconstructions can produce > 60 000 points, which with large initial scales leads to 200–300 Gaussians per pixel overlap — this creates a “fog field” in which training does not converge. Subsampling to ~16 000 points (hard-cap logic in the training engine) brings initial density to the level used by reference 3DGS, and dramatically reduces overlap. Set automatically with very dense SfMs; via CLI with `--max-points N`.

IN PLAIN WORDS

How many starting points from the camera reconstruction are used. With very dense reconstructions (more than 60 000) the app automatically limits to 16 000 — otherwise there is too much fog at the beginning. You don’t have to set this; the app handles it.

T30 cameraClusterOutlierMultiplier

DETAILS

Default: 10.0 (all presets — never overridden)
Range: 1.0 – 100.0 **Defined in:**

TECHNICAL

Multiplier for the camera cluster outlier filter, introduced in Phase 3.10 A.1. Before training the training engine computes the centroid of all camera positions and the maximum distance of any camera from the centroid. SfM points whose distance from the centroid exceeds `multiplier × maxCameraDistance` are discarded as outliers. Default 10× preserves the pre-Phase 3.10 behavior. A subtle bug: tighter SfM (cameras closer together) → smaller → smaller threshold → more points are discarded as outliers. Looser SfM → larger threshold → fewer points discarded. This is one of the causes of the Phase 3.9 funnel-vs-training anti-correlation: better SfM can downstream lead to worse training because too many initial points are killed. The field is available as a CLI override (`--camera-cluster-outlier-multiplier`) for the A.3 sweeps; not exposed in the UI. Values below 5 are usually too restrictive, above 20 ineffective.

IN PLAIN WORDS

Special filter that discards points from the reconstruction lying far away from the camera cloud. 10 = the app is generous, keeps almost everything. Increasing can make sense when far-away points (mountains in the distance) look in the image like floating blobs. Lower setting only in emergencies — you'll lose detail in the distance.

Regularization (T31–T37)

T31 coarseToFineBlurRadius

DETAILS

Default: 0 (= disabled) **Range:** 0 or 1 – 10 **Defined in:**

TECHNICAL

V369 experimental: box blur radius applied at the start of the densification phase to the ground truth image, reduced linearly to 0 by the end of densification (`T2`). Hypothesis: coarse-to-fine training — first learn coarse structures, then details — should yield more stable geometry. Tests: r=3 → 9.6 % worse, r=1 → 5.1 % worse. Reason for failure: densification decides based on image domain gradients, and blurring reduces exactly the signals important for "must clone here". Stays in the field catalog for future tests with a different density control scheme.

IN PLAIN WORDS

Experimental "coarse-first-then-detail" mode. Brought nothing, stays off.

T32 scaleRegWeight

DETAILS

Default: 0.0 (= disabled) **Range:** 0 or 0.0001 – 0.05 **Defined in:**

TECHNICAL

V370 experimental: L1 regularization on world-space scale. Penalizes Gaussians that grow too large — prevents "mega splats" covering whole wall surfaces with a single Gaussian. Tests: 0.01 → 200 % worse loss (2 M Gaussians, total explosion), 0.001 → 214 % worse. Reason: scale regularization conflicts with density control — smaller scales mean more Gaussians are needed, so density control splits more often, which in turn means more gradient work. Disabled, but documented for Mip-Splatting experiments (T74): in that context a scale lower bound could make sense.

IN PLAIN WORDS

Regularization that forces splats to stay small. Triggered splat explosions in tests (millions of splats). Do not enable.

T33 anisotropyRegWeight

DETAILS

Default: 0.0 (= disabled) **Range:** 0 or 0.0001 – 0.05 **Defined in:**

TECHNICAL

V445 experimental: penalty on the max(scale)/min(scale) ratio, intended to prevent extremely elongated “needle” Gaussians perceived as floaters. Tests: 0.01 → 69 % worse, 0.001 → 15 % worse. Reason: regularization forces splats toward “round” shape, which on a flat surface (wall, table, floor) is exactly wrong — there a flat, broad Gaussian is more efficient than a spherical one. Disabled. V549f offered with `T34 scaleRatioPruneThreshold` an alternative more targeted approach, which was also reverted.

IN PLAIN WORDS

Regularization that penalizes very long thin splats. Sounds sensible but was worse in tests. Off.

T34 scaleRatioPruneThreshold

DETAILS

Default: 0.0 (= disabled) **Range:** 0 or 5.0 – 100.0 (typically 10.0 – 30.0) **Defined in:**

TECHNICAL

Experimental post-training pruning that deletes each Gaussian whose max(scale)/min(scale) ratio exceeds the linear threshold set here. Targets extremely elongated “needle/disc” floaters that can’t be eliminated by regularization alone. In tests the pruning removed floaters as hoped, but also useful flat splats on walls and floors — the image became hole-ier. Hence off by default; the CLI flag (`--scale-ratio-prune N`) remains for targeted experiments. Recommended values if you do want to test: 30 (very conservative, removes only extreme outliers), 10 (aggressive, costs detail).

IN PLAIN WORDS

Attempt to filter out very elongated splats after training. Was net-negative in tests — floaters gone but also detail gone. Off.

T35 opacityRegWeight

DETAILS

Default: 0.0 (= disabled) **Range:** 0 or 0.0001 – 0.05 **Defined in:**

TECHNICAL

V446 experimental: binary cross-entropy penalty that pulls opacity toward 0 or 1 (i.e. away from "semi-transparent"). Hypothesis: sharper opacity distribution would improve image clarity. Tested combined with T33 → regularization costs quality, both disabled. Disabled. Attention: in 1.4.3 beta a bug appeared that had exactly this field with a changed default value (initializer = 0.01), which led to mass extinction of the Gaussian count (460 K → 5 in one iteration). Since 1.4.4 hard-pinned to 0.0 as default.

IN PLAIN WORDS

Regularization that makes splats either fully transparent or fully solid. Brings nothing, can even become dangerous (1.4.3 bug mass extinction). Leave at 0.

T36 opacityDecayFactor

DETAILS

Default: 0.0 (initializer = disabled), 0.9995 (`.full` , `.classicBalanced` — HTGS standard) **Range:** 0 (off) or 0.95 – 1.0 **Defined in:**

TECHNICAL

V546 implementation of the HTGS scheme (Hierarchical Time-Gating, Eurographics 2025): every `T37 opacityDecayInterval` iterations the sigmoid opacity of each Gaussian is multiplied by this factor. 0.9995 × 100 applications gives ~95 % residual per densification phase — a slight but steady downward pressure on all opacities, which reliably drops weakly contributing Gaussians below the `T14 pruneOpacityThreshold`. The result: 14 % better L1 loss on Horse Full (3-trial avg V546) vs V438 without decay. Active only during the densification phase (until `T2`), after that training continues without decay so the opacities established during refinement stay stable. Not used under MCMC (MCMC has its own mechanisms via `T67 mcmcRelocationInterval` + `T68 mcmcDeadOpacityThreshold`).

IN PLAIN WORDS

"Gentle fade" of all splats over the training time. Makes inactive splats transparent faster, so they disappear during cleanup. Was the most important quality lever of the V546 update: 14 % better. Built into the Quality preset. Even tweaking not recommended because precisely balanced.

T37 opacityDecayInterval

DETAILS

Default: 50 **Range:** 10 – 500 **Defined in:**

TECHNICAL

Iteration interval at which `T36 opacityDecayFactor` is applied. HTGS paper default 50, left at that in `.full`. Long intervals (>200) partially cancel the effect because between two applications enough gradient updates happen that opacity rises again. Shorter intervals (<20) make decay too aggressive. Active only in the densification phase.

IN PLAIN WORDS

How often the "fade" is applied. 50 = every 50 iterations a small fade step. Fits.

Refinement (T38–T44)

T38 gradientAccumulationSteps

DETAILS

Default: 1 (= "one view per Adam step") **Range:** 1 – 8 **Defined in:**

TECHNICAL

V424 feature: number of views whose gradients are accumulated before an Adam update runs. With `> 1` the app runs on a separate "unfused" backward project path that sums the gradients into a separate buffer; the final application scales by 1/N to keep magnitude constant. V424 tested 2-view → quality-neutral but 10 % slower (because unfused is more expensive than fused). Reverted for `.full` but deliberately used for MCMC — `.fullMCMC` runs with, but V544a tests showed that with the quality gap to Classic shrinks to 5 % (instead of 11 %). In the initializer default 1, in the current preset 1, remains a CLI flag (`--accum-steps N`).

IN PLAIN WORDS

How many images the app considers before adjusting the splats. 1 = each image individually. Higher = look at several images together and then apply an average. Brings nothing in the standard case; under MCMC 2 can help a bit.

T39 testViewIndices

DETAILS

Default: `[]` (= empty, all views are used for training) **Range:** `Set<Int>`, arbitrary subset of camera indices **Defined in:**

TECHNICAL

V546 feature: set of camera indices that are NOT used for training but saved as holdout for PSNR/SSIM/LPIPS evaluation. Set automatically when the `--benchmark` CLI flag is active: then every 8th view starting from index 0 (LLFF standard, identical to Mip-NeRF 360 and 3DGS paper conventions). Without benchmark empty — training uses all views. **Caution:** manually setting this field without understanding the indices can make the benchmark unusable (e.g. when all indices are set above N while there are only N-50 views → no holdouts → no evaluation). When you export your own preset, testViewIndices is not persisted because it is scene-dependent and would otherwise leave nonsensical values between different datasets.

IN PLAIN WORDS

Which images during training are "left out" to use later for quality measurement. You don't set this yourself; the `--benchmark` flag does it automatically (every eighth image is test). If you set your own indices: dangerous, can corrupt the benchmark.

T40 refinementPruneInterval

DETAILS

Default: 0 (= disabled) **Range:** 0 or 100 – 5 000 **Defined in:**

TECHNICAL

V425 feature: every N iterations during the refinement phase (after `T2`) an additional prune pass runs that removes Gaussians with sigmoid(opacity) < `T41 refinementPruneOpacityThreshold`. Reason: during densification there are regular density control calls, after that not anymore — but Gaussians whose opacity continues to drop stay in the buffer. V425 tested and reverted: the additional pruning correlated with V426 (Two-Phase Densification, which also ended in 0-Gaussians cascade failure). Disabled. CLI flag available for experiments; if enabled, 1 000 or 2 000 are sensible values.

IN PLAIN WORDS

Additional cleanup during the refinement phase. Brings nothing, stays off.

T41 refinementPruneOpacityThreshold

DETAILS

Default: 0.0 (= “use `T14`”) **Range:** 0 or 0.001 – 0.1 **Defined in:**

TECHNICAL

V425b: separate opacity threshold for refinement pruning. After densification most Gaussians have reached a clearly higher opacity (> 0.001), so the default `T14 pruneOpacityThreshold` would be too lax. If `T40` is active, this field determines its own threshold. At 0.0 `T14` is used as before. Only relevant if T40 > 0.

IN PLAIN WORDS

Threshold for the additional refinement cleanup (see T40). Both fields inactive, so irrelevant.

T42 midTrainingCompactificationIterations

DETAILS

Default: `[]` (= disabled) **Range:** `[Int]`, values in `(densifyUntilIteration, maxIterations)` **Defined in:**

TECHNICAL

V549 feature: explicit iteration points during the refinement phase at which a compactification pass runs (removes sigmoid(opacity) < 0.01 + outlier-scale Gaussians, same logic as `T56 postTrainingCompactification`). Reason: long refinement phases can show confetti/floater accumulation, whose SH then overfits to view-specific artifacts. Typical configuration if enabled: `[10000, 20000, 30000]` for 40K Classic. **BUT:** V549 A/B tests on the Family dataset showed worse L1 in all configurations: `[10K,20K,30K]@0.01` → –48 % count but +36 % L1; `[20K,30K]@0.005` → –44 % count but +45 % L1; `[20K,30K]@0.001` → –17 % count but +87 % L1. Hence disabled. CLI flag `--mid-compact "10000,20000"` available, if you prefer the visual floater tradeoff (less confetti in the viewport) over the loss regression.

IN PLAIN WORDS

Mid-training cleanup actions during training. In tests the cleanup made the end result worse (yes, fewer floaters, but also less detail). Off, can be enabled via CLI, in case floaters bother you more than a somewhat mushier image.

T43 frustumCullEnabled

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNICAL

V549b feature: after training all Gaussians outside the union of all training camera frusta are removed. Such Gaussians were never constrained by the loss signal and are always floaters. Particularly effective for scenes where the novel view sits behind or beside the camera path (e.g. behind a linear drone flight) — the floaters there are never visible during training but very much so later when moving in the 3D viewer. V549b A/B on drone flights showed positive results, hence available as opt-in. Default false because for object captures with full orbit coverage the frustum union encompasses the whole scene and the feature removes nothing — offered in Settings under "Floater Reduction" and also implicitly tested in Q9 Outdoor preset via `T44 frustumCullExpansion` (Q7 BayesOpt didn't enable it though, because outdoor sky dome solves the same problem better).

IN PLAIN WORDS

Special filter for drone flights or linear captures: after training, splats that were not "seen" in any camera are deleted. Optional, switchable on in Settings. For simple object captures unnecessary.

T44 frustumCullExpansion

DETAILS

Default: 1.1 **Range:** 1.0 – 2.0 **Defined in:**

TECHNICAL

NDC margin for `T43 frustumCullEnabled`. 1.0 would cut exactly at the image edge, which would trim wobbly splats near the edge too aggressively. 1.1 = 10 % padding beyond the exact camera framing — gives some tolerance for edge pixels that might still become visible in a slightly shifted novel view. Values > 1.2 make the cull practically ineffective because the expanded frustum encompasses too much space.

IN PLAIN WORDS

How strictly the above-mentioned filter clips. 1.1 = a bit of safety margin to the image edge. Leave the value.

Sky Dome (T45–T48)

T45 skyDomeEnabled

DETAILS

Default: false (initializer + all presets except P9 Outdoor) **Range:** boolean **Defined in:**

TECHNICAL

V549e feature: before training starts a spherical point cloud is generated (Fibonacci sphere with `T46` sample points), placed at a radius of `T47 skyDomeRadiusMultiplier × scene_extent` around the scene center, and initialized with the colors from sky-masked pixels of all training cameras (see `T20 skyMaskingEnabled`). These sky dome Gaussians are inserted at the beginning of the Gaussian buffer and during training "frozen" (position/scale/rotation gradients = 0, only SH and opacity remain optimizable). Effect: instead of black "confetti" areas in the distance, the user sees a real sky in novel views. The V549e MVP works very well on drone and landscape scenes; in P9 Outdoor preset default-on. Leave off for indoor scenes — the sphere would dangle uselessly outside the room.

IN PLAIN WORDS

Enables an artificial "sky dome" around the scene. Makes outdoor captures much nicer: instead of black blobs at the image edge, the app shows the real sky. Mandatory for drone flights and landscapes, pointless for interiors.

T46 skyDomeSampleCount

DETAILS

Default: 5 000 **Range:** 1 000 – 50 000 (typical 2 000 – 10 000) **Defined in:**

TECHNICAL

Number of Fibonacci sphere sample points on the sky dome sphere. Higher values → denser sky dome (better at large resolutions and lots of visible sky), but higher memory consumption. 5 000 is the sweet spot for 4K renderings; at lower resolutions 2 000–3 000 suffice. The points are initialized by cosine distance to each training camera view vector with the corresponding sky-masked pixels — sample points whose view cone is seen by no camera keep a low opacity initial value, but stay unchanged during training (frozen).

IN PLAIN WORDS

How dense the artificial sky is. 5 000 points usually suffice. More = smoother transition from the distance, but costs some memory.

T47 skyDomeRadiusMultiplier

DETAILS

Default: 30.0 (initializer + most presets), 59.0 (P9 Outdoor, Q7 BayesOpt optimum) **Range:** 5.0 – 200.0 **Defined in:**

TECHNICAL

Radius of the sky dome sphere relative to the scene extent (= mean distance between camera positions). 30 = the sphere has 30× the diameter of the camera cloud. Too small (< 5) → sky dome interferes with the scene itself (e.g. a sky dome splat lands in the foreground); too large (> 100) → float32 precision loss at the sky dome positions, which triggers render glitches in the distance. Q7 BayesOpt on Bicycle (Mip-NeRF 360) found 59.0 as scene-specific optimum for outdoor — this suggests that the default 30.0 is too small for deep landscapes and that sky dome pixels visibly render as a “wall” in image-edge regions.

IN PLAIN WORDS

How far away the artificial sky dome should be. 30 = quite far. With big landscapes 50–60 is better (Outdoor preset handles that automatically). Too small would be like having blobs right in front of the lens.

T48 frozenGaussianCount

DETAILS

Default: 0 (= no frozen Gaussians) **Range:** 0 or 1 – `T46` **Defined in:**

TECHNICAL

Number of Gaussians at the start of the buffer whose position/scale/rotation gradients are zeroed in the optimizer — they remain spatially rigid over the entire training. Density control may not clone, split or prune them. Used for sky dome injection (see `T45`): when sky dome is on, this field is automatically set to `T46 skyDomeSampleCount`. Manual setting is possible (e.g. to freeze a pre-placed point cloud from a LiDAR scan), but not directly accessible in the UI. Important: the first N Gaussians in the buffer are always the frozen ones — the order in the buffer decides, not an explicit index.

IN PLAIN WORDS

How many splats at the beginning are fixed and may not move. Set automatically to the sky dome count when sky dome is on. You don’t need to tweak this yourself.

Adam + LR Schedule (T49–T55)

T49 adamResetIteration

DETAILS

Default: 0 (= disabled) **Range:** 0 or 100 – **Defined in:**

TECHNICAL

V430 feature: iteration at which the Adam optimizer momentum accumulators (m1, m2) are reset to zero. Bias correction afterwards runs with `(iter - adamResetIteration)` instead of `iter`. V430 tested reset at 5 000 (after densification end) → 12.8 % worse loss. Reason: the Adam momentum that built up during densification carries information about the typical gradient magnitudes and accelerates the refinement phase. Throwing it away costs the first ~500 iterations of refinement in convergence. Disabled. Stays as CLI flag for research experiments.

IN PLAIN WORDS

Reset button for the internal Adam optimizer “memory”. Hurt in tests, stays off.

T50 positionLRScheduleEndIteration

DETAILS

Default: 0 (initializer = “use maxIterations”), 20 000 (`.full` — cosine ends at 20K although maxIter=35K), 30 000 (`.fullClassicPaper`) **Range:** 0 or 1 000 – **Defined in:**

TECHNICAL

V431 feature: iteration at which the cosine annealing curve for position LR reaches its minimum. If 0, this is identical to `T1 maxIterations`. If > 0, the schedule runs up to this value and stays constant at `T4 positionLearningRateFinal` afterwards. This allows an “extended refinement phase” with minimal but constant learning rate — refines positions slowly without a new decay. `.full` does this (schedule ends at 20K, training runs to 35K), V434c/V434d confirmed: 15K and 25K both about the same, 20K marginally optimal. Used together with `T51` to also modify the non-position LRs in the extended phase.

IN PLAIN WORDS

When the app stops lowering the position learning rate further. If lower than the maximum iteration, the rest runs with a constant mini rate — this refines very slowly but very stably. Built into the Quality preset, you don’t need to tweak.

T51 extendedPhaseLRDecay

DETAILS

Default: 0.0 (= disabled, constant LRs) **Range:** 0 or 0.01 – 1.0 **Defined in:**

TECHNICAL

V433 feature: minimal multiplier for the non-position LRs (scale, rotation, opacity, SH) in the "extended phase" — i.e. after `T50` is reached and position LR is already at `T4`. If 0.1, scale/rotation/opacity/SH are themselves cosine-decayed from 1.0 (= their standard LR) to 0.1× of their standard. If 0.0 (default), they stay constant. V457 tested full decay (0.0, i.e. decay to zero) against no decay and found: avg 0.0400 (2 runs), the same loss as V438 without decay. Behavior cleaner with decay but not measurably better. Hence disabled. Stays in the CLI as `--nonpos-lr-scale F`.

IN PLAIN WORDS

In the late refinement phase also make color and shape learning rates smaller. Makes training "more stable" but empirically not better. Off.

T52 adaptiveDensifyThreshold

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNICAL

V440 experimental: when true, the app computes each densification step the p98 of the current gradient distribution and uses it as dynamic threshold (clamped to at least 0.5× of the configured value from `T11` so it doesn't drift too far). Hypothesis: automatic adaptation to the current scene phase would make density control more robust — e.g. stricter pruning at the start, laxer later, or vice versa. V440 tested and reverted: catastrophic drop to 63 K Gaussians (mass pruning, because the p98 in the first iterations is extremely high and afterwards almost nothing exceeds the threshold). The fixed threshold is already well calibrated, dynamic adjustment hurts more than it helps. Q5 (T77) offers an alternative adaptive logic via rolling median that avoids the problem.

IN PLAIN WORDS

Adaptive version of the densify threshold. In tests catastrophic (splat count crashed to 63K). Off. Q5 has a better variant.

T53 mergeAfterDensification

DETAILS

Default: false (initializer), true (`.full` , `.classicBalanced` , `.fullClassicPaper`) **Range:** boolean **Defined in:**

TECHNICAL

V438 feature: at the end of the densification phase (iter `T2`) a one-time merge pass runs that combines nearby Gaussians with similar scale and color. Reduces the Gaussian count by typically 5–15 % without visible quality loss. Reason: after intense cloning, clusters of nearly identical Gaussians arise that contribute nothing new — merging frees optimizer capacity for other areas. Default in Classic Quality presets. Not used under MCMC, because MCMC through its relocation logic doesn't let such clusters form in the first place.

IN PLAIN WORDS

At the end of the splat duplication phase combine clones that are nearly identical. Reduces data size without visible effect. By default on in the Quality preset.

T54 densifyPhase2FromIteration

DETAILS

Default: 0 (= disabled) **Range:** 0 or `T2` – `T1` **Defined in:**

TECHNICAL

V426 experimental: enables a second densification phase that starts after the refinement pause at this iteration and runs until `T55` . Hypothesis: after a refinement phase the gradient accumulators have more stable magnitudes and can more precisely indicate which regions still need additional Gaussians. V426 tested and reverted: two-phase densification fell into 0-Gaussians cascade failure (combined with V425 refinement pruning it destroyed the buffer). Disabled. CLI flag available for experiments.

IN PLAIN WORDS

Second multiplication round after a pause. In tests it annihilated the splat inventory. Off.

T55 densifyPhase2UntilIteration

DETAILS

Default: 0 **Range:** 0 or `T54` – `T1` **Defined in:**

TECHNICAL

End of V426 two-phase densification. Only relevant when `T54 > 0`. Both fields together disabled.

IN PLAIN WORDS

End of the second multiplication round (see T54). Both off.

Post-Processing + Apple AI (T56–T60)

T56 postTrainingCompactification

DETAILS

Default: true (in all production presets), false (`.quickTest` , `.preview`) **Range:** boolean **Defined in:**

TECHNICAL

V443 feature: after training ends, Gaussians with sigmoid(opacity) < 0.01 are hard-removed (they practically don’t contribute to the image anymore). Reduces Gaussian count by typically 58 % and export file size by 55 % without visible quality loss. On by default in production presets — the end result should ship as compact as possible. Off in `.quickTest` , because a diagnostic run isn’t exported anyway. Unlike `T42 midTrainingCompactificationIterations` (V549) the compactification only happens at the end — refinement can use all Gaussians until then.

IN PLAIN WORDS

Cleanup after training: nearly invisible splats are removed. Makes the export file roughly half as large without quality loss. Mandatory feature, leave off only in diagnostic runs.

T57 metalFXUpscaling

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNICAL

V444 feature: enables Apple's MetalFX Spatial Upscaler instead of bilinear interpolation in the 3D viewer output. When training resolution < viewport size (e.g. training at 0.5×, viewport display at full resolution), MetalFX can deliver a clearly sharper image. Changes live in the viewport, no retraining required. Mutually exclusive with `T58 mpsLanczosScaling` — MetalFX takes precedence. Recommendation: enable when the image in the viewer looks "washed out" compared to expected detail.

IN PLAIN WORDS

Apple ML-based image sharpening in the 3D viewer. Helps when you trained at a lower resolution and want to show the result in full screen. Live toggle, give it a try.

T58 mpsLanczosScaling

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNICAL

V444 feature: MPSImageLanczosScale for viewport scaling instead of bilinear interpolation. Lanczos is a classic Sinc-based resampling method that delivers significantly sharper results than bilinear with minimal overhead. Live toggle. Overridden by `T57` when both are on.

IN PLAIN WORDS

Classic sharpening method for the 3D viewer (Lanczos). MetalFX (T57) is ML-based and usually better; Lanczos is a less aggressive alternative.

T59 livePreviewInterval

DETAILS

Default: 50 (initializer and most presets) **Range:** 0 (off) or 10 – 5 000 **Defined in:**

TECHNICAL

How often during training the 3D viewer is refreshed with the current Gaussians. 50 = every 50 iterations a new render in the viewer — enough to observe progress without slowing training. 0 = viewer is not updated at all (background training, max speed). Typical adjustment: under `.quickTest` drop to 10 (you want to see every step), on long MCMC runs raise to 500–2000 (update overhead in sum noticeable).

IN PLAIN WORDS

How often the 3D preview during training is refreshed. 50 = every 50 iterations. Higher = less often = a bit faster, but you see progress less often. 0 = no preview (for maximum speed).

T60 perceptualLossWeight

DETAILS

Default: 0.0 (= disabled) **Range:** 0 or 0.001 – 0.5 **Defined in:**

TECHNICAL

V444 future feature: weight of a perceptual loss term via MPSGraph (VGG-like small network). Would capture structural and textural similarity at a higher semantic level than L1+SSIM — typical in research pipelines where "pixel-perfect" matters less than "looks realistic". Implementation pending (code stub present, but forward pass not implemented). Default 0.0. Stays in the field catalog for future activation; CLI flag `--percep-weight F` reserved.

IN PLAIN WORDS

Planned feature that with AI assistance aims for "natural looking" instead of "pixel-accurate". Not yet fully implemented.

MCMC Densification (T61–T73)

T61 densificationStrategy

DETAILS

Default: `.classic` (initializer + Classic presets), `.mcmc` (all MCMC presets + Scene-Class) **Range:** `.classic` or `.mcmc` **Defined in:**

TECHNICAL

Chooses between classic densification (clone/split/prune, Kerbl et al. 2023) and MCMC densification (Stochastic Gradient Langevin Dynamics with relocation, Kheradmand et al. NeurIPS 2024). With `.classic` `T11–T16` is evaluated, with `.mcmc` the `T62–T73`. Beware when switching: Classic defaults and MCMC defaults are completely differently calibrated — whoever flips the picker in the Expert View without loading a matching preset risks 1.4.3-bug-style mass extinction (460 K → 5 in one iteration, because MCMC opacity reg at 0.01 kills the Classic opacities). Hence the MCMC initializer defaults are deliberately “soft-washed” (all reg values 0.0).

IN PLAIN WORDS

Which algorithm is used to multiply the splats. Classic = original method (fast, many splats). MCMC = newer method (slower, far fewer splats, but more compact). The presets pick the right one. Only switch yourself if you also load the matching preset (P5–P7 or P8–P10).

T62 mcmcMaxGaussians

DETAILS

Default: 150 000 (initializer + `.fullMCMC` + `.mcmcBalanced`), 100 000 (`.mcmcPreview`), 1 500 000 (`.fullMCMCMip` — Mip-Splatting variant with 10× budget), 1.19 M (`.renderPreset`), 1.25 M (`.outdoorPreset`), 670 K (`.indoorPreset`) **Range:** 0 (= "use buffer capacity") or 10 000 – 5 000 000 **Defined in:**

IN PLAIN WORDS

Maximum number of splats under MCMC. 150 000 is the default and suffices for most scenes. Outdoor and render presets (P8, P9) go to 1+ million for more detail-rich scenes. Raising can bring detail, costs memory; lower is more an emergency brake.

TECHNICAL

Hard upper bound for the number of Gaussians under MCMC strategy. The count grows gradually by `T70 mcmcGrowthRate` (typically 5 %) per relocation step up to this cap. V473/V531 found 150 K as sweet spot — above 200 K dilutes splat quality (too many small redundant Gaussians), below 100 K leaves the scene under-densified. With very large scenes (e.g. 1 545-photo drone flight with 158 K SfM init), 150 K is too low — hence the 1.4.5 extension `T72 mcmcCapMultiplier` + `T73 mcmcAutoScaleByScene`. Q7 BayesOpt found scene-specific optima between 670 K (Indoor) and 1.25 M (Outdoor). With value 0 the engine uses the full buffer capacity as cap.

T63 mcmcNoiseScale

DETAILS

Default: 0.00005 (5e-5 = paper default) **Range:** 1e-6 – 1e-3 **Defined in:**

IN PLAIN WORDS

How much random "wobble" the app allows the splats so they can find the best spot themselves. The default value is optimal in tests. If you crank this up, the splats get restless.

TECHNICAL

Multiplier for the Gaussian noise that in each MCMC iteration is added to the position of each Gaussian (SGLD logic). Higher = more exploration (Gaussians wander more, finding potentially better spots), lower = more exploitation (Gaussians stay where they're already good). V467 and V536 confirmed 5e-5 as optimal — 1e-5/2e-5 too little exploration, 1e-4 too much (splats diffuse). Cosine-decayed over training time until `T69 mcmcNoiseDecayEnd` — at the end of the decay range noise is effectively 0 and the Gaussians converge.

T64 mcmcOpacityRegWeight

DETAILS

Default: 0.0 (= disabled in RadianceKit defaults, paper: 0.01) **Range:** 0 or 0.001 – 0.05 **Defined in:**

TECHNICAL

MCMC-specific L1 penalty on opacity. Paper default 0.01 (pushes unused Gaussians toward zero, makes them available for relocation). V464b showed however: without reg it's measurably better in RadianceKit (Session 28 confirmed). Reason: the pruning criterion defined with `T68 mcmcDeadOpacityThreshold` is enough on its own — an additional L1 penalty also forces valuable low-opacity Gaussians to die. Hence default 0. **Caution:** in the 1.4.3 beta build the initializer default was mistakenly 0.01, which resulted in the mass extinction bug (see T61 explanation); since 1.4.4 fixed at 0.0.

IN PLAIN WORDS

MCMC special regularization. Off because the other MCMC mechanism (threshold in T68) already covers this. Leave at 0.

T65 mcmcScaleRegWeight

DETAILS

Default: 0.0 (= disabled, paper: 0.01) **Range:** 0 or 0.001 – 0.05 **Defined in:**

TECHNICAL

MCMC-specific L1 penalty on the scale eigenvalues. Paper default 0.01. V464b: without reg better, same reasoning as T64. Disabled in all RadianceKit MCMC presets. Caution as with T64: 1.4.3 bug.

IN PLAIN WORDS

Like T64 but for splat size. Off.

T66 mcmcRelocationInterval

DETAILS

Default: 100 (initializer + all MCMC presets, paper standard), 155 (P9 Outdoor — Q7 BayesOpt optimum) **Range:** 50 – 500 **Defined in:**

TECHNICAL

Iteration interval at which MCMC relocates dead Gaussians (sigmoid(opacity) < `T68 mcmcDeadOpacityThreshold`) to new positions. V537 tested 50 (too disruptive, loss fluctuates) and 200 (marginally worse, MCMC loses responsiveness). 100 is optimal. Q7 BayesOpt on Bicycle found 155 as scene-specific optimum for outdoor — the slightly longer intervals give Adam more time to integrate newly placed Gaussians before the next reloc event puts them under pressure.

IN PLAIN WORDS

How many iterations between MCMC moving dead splats elsewhere. 100 is standard. You don't need to tweak this yourself — Outdoor preset already has the optimal value.

T67 mcmcWarmupIterations

DETAILS

Default: 500 **Range:** 100 – 5 000 **Defined in:**

TECHNICAL

Number of initial iterations during which no MCMC relocation happens. Only after this warmup does the reloc logic kick in. Reason: in the first iterations the opacity values have not yet stabilized — if reloc started directly, Gaussians would be placed at the wrong spots and have to be moved again right away, which would destroy Adam momentum. Paper default 500. RadianceKit adopts this value because V464b showed it is robust.

IN PLAIN WORDS

How many iterations MCMC is allowed to first "arrive" before starting to move splats. 500 is standard and fits.

T68 mcmcDeadOpacityThreshold

DETAILS

Default: 0.005 (initializer, paper standard), 0.01 (`.fullMCMC` and all MCMC presets — V535 optimum) **Range:** 0.001 – 0.05 **Defined in:**

TECHNICAL

sigmoid(opacity) threshold below which a Gaussian counts as "dead" and is eligible for relocation. V535 found 0.01 as optimal (0.005 marginal, 0.02 worse). Higher = more aggressive reloc (more Gaussians moved), lower = more cautious. 0.01 corresponds to roughly "0.5 % visual visibility". P10 Indoor uses 0.0142 as optimum via Q7 BayesOpt.

IN PLAIN WORDS

Below which transparency a splat counts as "dead" so MCMC pushes it elsewhere. 0.01 is optimal in our tests. You don't need to tweak this yourself.

T69 mcmcNoiseDecayEnd

DETAILS

Default: 0 (initializer = "no decay"), 160 000 (`.fullMCMC` = 80 % of 200K), 96 000 (`.mcmcBalanced` = 80 % of 120K), 40 000 (`.mcmcPreview`) **Range:** 0 or 1 000 – **Defined in:**

TECHNICAL

Iteration at which the `T63 mcmcNoiseScale` noise is fully damped to zero (cosine decay from iter 0 to here). V497c/V502 found 80 % of maxIterations optimal — gives MCMC enough exploration time but leaves the last 20 % to convergence without noise. 0 = constant noise over all iterations (rarely sensible, MCMC can't converge then).

IN PLAIN WORDS

When the random "wobbling" of splats stops. In the MCMC presets at 80 % of total iterations — first exploration, then convergence. Leave the value.

T70 mcmcGrowthRate

DETAILS

Default: 0.05 (paper standard = 5 %) **Range:** 0.01 – 0.2 **Defined in:**

TECHNICAL

Growth rate of the MCMC population target per relocation step. The logic: at each reloc event the target population size is multiplied by `(1 + growthRate)`, until `T62 mcmcMaxGaussians` (or the variant scaled via `T72/T73`) is reached. V512/V522 found 0.05 as optimal — higher values lead to too-fast growth (Gaussians are inserted before Adam momentum can integrate them), lower values to under-densified scenes at the end.

IN PLAIN WORDS

How fast the splat count grows under MCMC. 5 % per step is optimal. Leave the value.

T71 mcmcSigmoidK

DETAILS

Default: 100.0 **Range:** 10.0 – 500.0 **Defined in:**

TECHNICAL

Sigmoid sharpness parameter for the MCMC noise attenuation. In the SGLD step the per-Gaussian noise is damped by — high-opacity Gaussians (whose logit is positive) get exponentially less noise than low-opacity ones. K = 100 is sharp, meaning the transition from “full noise” to “no noise” happens very quickly around opacity 0.5. V484–V487 found K = 100 optimal — smaller values (10–50) also let high-opacity Gaussians wobble (destroys converged Gaussians), larger ones (> 500) make the transition artificially hard and dead Gaussians don’t get moved at all anymore.

IN PLAIN WORDS

Special parameter that determines how sharply MCMC distinguishes between “transparent enough to move” and “solid, don’t touch”. The default is optimal. Don’t tweak.

T72 mcmcCapMultiplier

DETAILS

Default: 3.0 (initializer + `.fullMCMC`), 2.0 (`.mcmcPreview`), 2.5 (`.mcmcBalanced`), 2.98 (P8 Render), 5.32 (P9 Outdoor), 1.76 (P10 Indoor)
Range: 0 (= disabled) or 1.0 – 10.0 **Defined in:**

TECHNICAL

1.4.5 feature: scene-adaptive cap scaling. When `T73 mcmcAutoScaleByScene` is true, the effective cap is computed as (clamped to buffer capacity). Background: with large scenes (e.g. 1 545-photo drone flight → 158 K SfM init) `T62 = 150 000` is too low — density control couldn't grow at all. With multiplier 3.0 the cap is scaled to 474 K in this example (158 K × 3.0). Q7 BayesOpt found scene-specific optima: outdoor benefits from a high multiplier (5.32 → ~830 K cap with 156 K bicycle init), indoor is happy with 1.76 (walls saturate faster). For the complete resolution of the cap see the method.

IN PLAIN WORDS

Multiplier that automatically adapts the splat cap to scene size. Large scene = more starting points = higher cap. The default 3× fits for most scenes; Outdoor preset goes to 5× (large depth ranges), Indoor to 1.76× (walls bound things anyway).

T73 mcmcAutoScaleByScene

DETAILS

Default: true (initializer + all MCMC presets)
Range: boolean **Defined in:**

TECHNICAL

1.4.5 feature: master switch for the scene-aware cap logic (see T72 +). When false, only `T62 mcmcMaxGaussians` is used as cap (back to 1.4.4 behavior). On by default because the mass extinction problems with large scenes from 1.4.3 would otherwise return. Manually disable only when you explicitly want to set a hard cap — e.g. to train a 150 K variant whose final size is planable.

IN PLAIN WORDS

Enables automatic adjustment of the splat cap to scene size. On by default. Leave off only when you want exactly a specific splat count yourself.

Mip-Splatting (Q1.5) (T74–T76)

Status: Q1.5 was rejected on 2026-05-25 after 14 autonomous iterations + overnight 1.5M confidence check as "closed no-win" (max Δ@2× = +0.27 dB, original gate required ≥ +1.5 dB mean over 0.5×/2×, FAILS on 0/11 pair scenes). The

fields remain **opt-in** for research experiments; all production presets have them off. See verdict: `docs/plans/2026-05-25-phase-q1.5-final-verdict.md`.

T74 useMipSplatting

DETAILS

Default: false (all production presets), true (`.fullMCMCMip` — research sibling) **Range:** boolean **Defined in:**

TECHNICAL

Enables Mip-Splatting (Yu et al. CVPR 2024): 3D smoothing filter + 2D filter + α compensation that limits per-Gaussian frequency to the Nyquist bound of the densest training camera sampling rate. Theoretical goal: eliminate aliasing when rendering at off-training scales (0.5× or 2× of training resolution). Enabled in the preprocess and backward projection shaders, functional correctness verified in Q1.5-D test. But: the original acceptance gate (Δ@1× ≥ +0.3 dB AND avg(Δ@0.5×, Δ@2×) ≥ +1.5 dB) was reached on none of 11 pair scenes. Maximum observed: family 750K classic Δ@2× = +0.270 dB. Outdoor scenes (Truck, Flowers) even showed worsening 1× and 0.5×. Hypothesis: 3D smoothing competes with MCMC relocation at high Gs. Field remains for future multi-scale re-eval with correct Mip-NeRF-360 methodology (see O3 backlog in the benchmark path).

IN PLAIN WORDS

Aliasing filter from a 2024 paper. Theoretically great, practically it brought nothing in our tests and sometimes even hurt. Remains available for tinkerers, but we don't recommend it. Leave off.

T75 mipSmoothing3DScale

DETAILS

Default: 0.2 (paper default) **Range:** 0.05 – 1.0 **Defined in:**

TECHNICAL

3D smoothing scale parameter (Yu et al. §3.3, paper default 0.2). Larger = more world-space smoothing per Gaussian (= more anti-aliasing but also more blur at the default scale), smaller = sharper but more aliasing-prone. Only consulted when `T74 useMipSplatting = true`. Not further optimized in Q1.5 tests — the A/B gate already lost with the paper default 0.2, further sweeps would be pointless.

IN PLAIN WORDS

. If you haven't enabled Mip, irrelevant.

T76 mipFilter2DVariance

DETAILS

Default: 0.3 (= exactly the V242 legacy behavior)
Range: 0.1 – 1.0 **Defined in:**

TECHNICAL

2D Mip filter variance added to the Σ_2D diagonal (variance directly, not squared). 0.3 is exactly the V242 legacy value that was hardcoded in the kernel before Mip-Splatting. When `T74 useMipSplatting = false`, the kernel ignores this value completely and writes the hardcoded 0.3 — so that the baseline cannot regress (Codex round 1 S3-1 guarantee). When, the value set here is used. Stays in the field catalog for Mip sweeps.

IN PLAIN WORDS

Further Mip-Splatting parameter. With Mip off: irrelevant.

Adaptive Densification (Q5) (T77–T79)

T77 adaptiveDensification

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNICAL

Q5 feature: rolling-median tracker as alternative to the fixed `T11 densifyGradThreshold`. When true, in every densify step the current threshold is overwritten with `median(last N avgGrad samples) × T79 adaptiveDensifyMultiplier`. N = `T78 adaptiveWindow`. Stricter than V440 p98 (the catastrophic 63 K pruning trap), median + 2× sits around the p70–p80 of the gradient distribution in steady state. Q5 tests: alone FAIL 0/3 scenes, but together with Q6 (see T80/T81) PASS 1/3 scenes — the Q5+Q6 bundle passed on 2026-05-25 as opt-in and is enabled via CLI `--adaptive-densify`. Q6 is the "carrier" of the quality gain, Q5 contributes more to stability.

IN PLAIN WORDS

Self-learning densify threshold. Instead of a fixed sensitivity setting, the app adapts to the scene. On its own not better in tests, but together with the curriculum from Q6 yes. Either both on or both off.

T78 adaptiveWindow

DETAILS

Default: 1 000 **Range:** 100 – 10 000 **Defined in:**

TECHNICAL

Rolling median window in densification events (NOT iterations — each `T13 densifyInterval` step yields one sample). Default 1 000 — with that means the last 100 000 training iterations contribute to the median, so typically the entire training history up to here. Early phase (before `T78` samples): tracker returns nil → fallback to fixed threshold `T11`. Only relevant when.

IN PLAIN WORDS

How many old densify steps feed into the median for T77. Default 1000 is good. Only relevant when Q5 adaptive is on.

T79 adaptiveDensifyMultiplier

DETAILS

Default: 2.0 **Range:** 1.0 – 4.0 **Defined in:**

TECHNICAL

Multiplier on the rolling median for the adaptive threshold. Default 2.0 corresponds to roughly p70–p80 of the typical gradient distribution. Lower = more aggressive growth (more clones), higher = stricter (fewer clones). Q5 tests in range 1.5–3.0 — 2.0 best default. Only relevant when.

IN PLAIN WORDS

Factor for T77/T78. Default 2.0 = stricter than the typical median. Don't tweak.

Curriculum (Q6) (T80–T81)

T80 curriculumResolutionRamp

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNICAL

Q6 feature: training resolution starts at 0.5× and switches at `T50 positionLRScheduleEndIteration / 2` (or `T1 maxIterations / 2`, if T50 is not set) to `T22 trainingRenderScale`. Uses the `resize/restoreImageBuffers` infrastructure developed in Q1.5.1. Overrides `T23 resolutionWarmupScale` when enabled. Q6 passed as "carrier of the quality gain" in the Q5+Q6 bundle (see T77) — the gradual resolution increase gives the app time to find coarse geometry at lower resolution before moving on to fine detail work. Via CLI: `--curriculum-resolution`.

IN PLAIN WORDS

"First coarse, then fine" for the training resolution. Half resolution in the first half, then full resolution. Helps in certain scenes, not in others — best to enable together with T81.

T81 curriculumSHProgression

DETAILS

Default: false **Range:** boolean **Defined in:**

TECHNICAL

Q6 feature: overrides `T21 shDegreeUpgradeIterations` with `[maxIter/4, maxIter/2, maxIter*3/4]`, distributing SH upgrades evenly across training time instead of front-loading them. Hypothesis: stable geometry is established before color detail explosion, which places the view-direction-dependent gloss effects more precisely. Q5+Q6 together PASS 1/3 scenes, Q6 as carrier of the gain (Q5 alone FAIL). Via CLI: `--curriculum-sh`.

IN PLAIN WORDS

"First shape, then color" — the gloss effects are unlocked only late in training, so the splats first find their position and size. Can be enabled together with T80; alone it doesn't bring quite as much.

Static Presets (TP1–TP9)

Only the structural differences from the initializer default here. The full marketing description of the eleven UI presets P1–P11 is in Chapter 7.

TP1 `.preview`

DETAILS

Diagnostic/preview preset for systems ≥ 10 GB RAM. Overrides vs. initializer: - 30 000 → 5 000 - 15 000 → 3 500 (70 % of maxIter) - 1.6e-6 → 1.6e-5 (10× higher, less aggressive decay) -,,,, each 2× (V176) - 3 000 → 100 000 (effectively off, V172: reset destroys short trainings) - `[1K, 2K, 3K]` → `[1K, 2K]` (V182: degree 3 doesn't converge in 2K iter) - 1.0 → 0.5

IN PLAIN WORDS

any initial assessment of a newly imported image series — 2–3 min wait, after which the result is enough for a binary "is a Quality run worth it?".

TP2 `.full`

DETAILS

Production Quality Classic. Overrides: - 30 000 → 35 000 (V550: 40K tests Truck overtraining +10.7 % Gs at –1.3 % L1) - 15 000 → 5 000 (V310 sweet spot, V338 7K worse) - All LRs 2× (V188) - 1.6e-6 → 1.6e-5 (V45 10×) - 2e-6 → 1.1e-6 (V335) - 100 → 200 (V112) - 0.005 → 0.001 (V393) - 3 000 → 100 000 (V194 disabled, V421 confirmed) - `[1K, 2K, 3K]` → `[2K, 5K, 8K]` (V228 delayed) - 0.0 → 0.9995 (V546 HTGS, 14 % improvement) - 50 (unchanged, V546) - false → true (V438) - 0 → 20 000 (V431) - true (V443, already initializer default for `.full`)

IN PLAIN WORDS

any standard photo capture (object, small room, sculpture) with < 500 images. The 14 % loss improvement announced in V546 vs V438 was confirmed as 3-trial average on Horse Full.

TP3 `.fullClassicPaper`

DETAILS

Q1.5-A test sibling of TP2, paper-faithful Classic. Overrides vs. TP2: - 35 000 → 30 000 (paper standard) - 5 000 → 15 000 (paper: 50 % of maxIter) - 1.6e-5 → 1.6e-6 (paper default) -,, back to paper defaults (0.05, 0.005, 0.001) - 1.1e-6 → 2e-7 (calibrated for ~1-2M Gs on Bicycle) - 200 → 100 (paper) - 0.001 → 0.005 (paper default) - 100 000 → 3 000 (paper §5.2, risky — can trigger V194 regression) - 0.9995 → 0.0 (paper has no decay) - 20 000 → 30 000 (cosine runs to 100 % of maxIter)

IN PLAIN WORDS

Q1.5 research experiments that need paper-magnitude Gaussian budgets (1–2 M) for Mip-Splatting tests. After the Q1.5 "closed no-win" verdict the preset remains accessible for advanced users but is not production-recommended.

TP4 `.fullMCMC`

DETAILS

Production Quality MCMC. Overrides vs. initializer: - 30 000 → 200 000 (V534, MCMC needs 5× more iter than Classic) - 15 000 → 160 000 (V504b 80 % of maxIter) - 1.6e-6 → 1.6e-5 - LR schedule like TP2 (all 2×) - 0.2 → 0.05 (V521b/V534: MCMC needs stronger L1 signal) - `[1K, 2K, 3K]` → `[2K, 5K, 8K]` - `.classic` → `.mcmc` - 150 000 (already in initializer, confirmed in preset) - 5e-5 (V467/V536 optimal) - 0.005 → 0.01 (V535 optimal) - 0 → 160 000 (80 % of maxIter, V497c/V502) - 3.0 (already in initializer) - true (already in initializer) - 3 000 → 200 000 (effectively off, MCMC uses reloc instead of reset)

IN PLAIN WORDS

Web delivery, object captures with detail aspirations, drone flights (although then P9 Outdoor is even better). 71 % fewer Gaussians than Classic at comparable L1.

TP5 `.fullMCMCMip`

DETAILS

Q1.5-D test sibling of TP4, with Mip-Splatting + paper-magnitude MCMC budget. Overrides vs. TP4:

- mcmcMaxGaussians 150 000 → 1 500 000 (10×, paper magnitude)
- useMipSplatting false → true (Mip on)

IN PLAIN WORDS

All other fields identical to TP4. Q1.5 D-PASS on Bicycle 2026-05-24 (breaks 12-iter multi-scale FAIL streak). Q1.5 final verdict 2026-05-25 still closed-no-win — Mip-Splatting gain not reproducible across 11 pair scenes. Preset remains opt-in.

TP6 `.classicBalanced`

DETAILS

Mid-tier Classic. Overrides vs. TP2: - 35 000 → 20 000 (V149: 20K = 30K at 33 % less time) - 20 000 → 0 (cosine runs to maxIter = 20K, no extended phase)

IN PLAIN WORDS

Standard cases with shorter wait time. V149 identified as sweet spot.

TP7 `.mcmcPreview`

DETAILS

MCMC diagnostic. Overrides vs. TP4: - 200 000 → 60 000 (V494b) - 160 000 → 48 000 (80 %) - 150 000 → 100 000 (V473b) - 160 000 → 40 000 (V494b) - 3.0 → 2.0 (1.4.5: Preview = lighter scaling)

IN PLAIN WORDS

Quickly see an MCMC result to judge whether TP4 or a Scene-Class preset is worth it.

TP8 `.mcmcBalanced`

DETAILS

Mid-tier MCMC. Overrides vs. TP4: - 200 000 → 120 000 (V518) - 160 000 → 96 000 (80 %) - 160 000 → 96 000 (80 %) - 3.0 → 2.5 (between Preview 2.0 and Full 3.0)

IN PLAIN WORDS

MCMC without the full 200K run. ~120 K iterations are a good compromise between quality and wait time.

TP9 `.quickTest`

DETAILS

Pure functional test. Overrides vs. initializer: - 30 000 → 1 000 - 15 000 → 500 - 2e-6 → 4e-6 (calibrated for 0.25× resolution) - 100 → 50 - 3 000 → 100 000 (off, since way too short) - 1.0 → 0.25

IN PLAIN WORDS

Sanity check “does training even start sensibly?”. Duration < 30 s on M3 Ultra. Guaranteed to look mushy.

Method:

Signature: `public func resolveMcmcMaxGaussians(initialPointCount: Int, bufferCapacity: Int) -> Int` **Defined in:**

TECHNICAL Single source of truth for the question “how many Gaussians can MCMC be allowed to grow to?” Computed from three inputs: the configured `T62 mcmcMaxGaussians` (with mass extinction floor 150 000 if 0), the (number of SfM init points) and the (preallocated Gaussian buffer size). Logic:

1. `base = T62 > 0 ? T62: 150_000` (the mass extinction floor protects against initializer default bugs like the 1.4.3 mass extinction incident)
2. If `T73 mcmcAutoScaleByScene && initialPointCount > 0 && T72 mcmcCapMultiplier > 0`:
   - `scaled = max(base, ceil(initialPointCount × T72))` else
3. If `bufferCapacity > 0`: `return min(scaled, bufferCapacity)`
4. Else `return scaled`

Example: Bicycle (Mip-NeRF 360, 194 photo frames) → SfM init ~156 K points, `T62 = 150 000`, `T72 = 5.32`,, buffer capacity 8 M. Resolved cap = `min(8M, max(150K, ceil(156K × 5.32))) = min(8M, 830K) = 830 K`. That is the effective growth cap the MCMC relocation logic adheres to.

📍 Computes the actual maximum splat count under MCMC. Takes your setting, looks at how many points your scene starts with, and scales by the `Multiplier`, if automatic adaptation is on. So the cap adapts to the scene instead of forcing the same value for a tiny and a huge scene. You don't have to call the method yourself — the training uses it internally.

Which field for what? (Cheat Sheet)

Goal	Fields to tweak
More detail in the distance	`T62 mcmcMaxGaussians` up, `T72 mcmcCapMultiplier` 5+
More detail overall (Classic)	`T1 maxIterations` up (≤ 40K), `T2 densifyUntilIteration` ≤ 14 % of T1
Reduce floaters in drone flights	`T43 frustumCullEnabled` on, `T20 skyMaskingEnabled` on, `T45 skyDomeEnabled` on
Nice sky in outdoor scenes	`T45 skyDomeEnabled` on, `T47 skyDomeRadiusMultiplier` 30–60
Smaller export file	Strategy `.mcmc` (T61), `T56 postTrainingCompactification` on, `T62 mcmcMaxGaussians` ≤ 200K
Faster training	`T22 trainingRenderScale` 0.5, `T1 maxIterations` halved — but not both!
Better highlights	`T21 shDegreeUpgradeIterations` with `[2K, 5K, 8K]` (no early front-load), MCMC + 200K iter
Keep Mac responsive	`T25 throttleDelayMs` 5–10 (costs ~15 % training time)
Live preview more often	`T59 livePreviewInterval` down to 10–20
Smoother transitions in shadows	`T17 ssimWeight` slightly up (0.15–0.25), but not above 0.3
Keep interiors compact	P10 Indoor preset (, `T72 = 1.76`)

Dangerous Fields

These fields can, with misconfiguration, lead to OOM, app crash, mass extinction of Gaussians, or unusable benchmark data. Handle with care:

- `T11 densifyGradThreshold` — halving can create 2–4× as many Gaussians, quickly blowing up GPU memory. Also note: must match the `T22 trainingRenderScale` (1.0× → 1e-6, 0.5× → 2e-6, 0.25× → 4e-6).
- `T72 mcmcCapMultiplier` — with large scenes with > 200 K SfM init points and a multiplier > 5 a resolved cap of millions of Gaussians arises. On 36 GB RAM Macs OOM is possible. Outdoor preset 5.32 works only because Mip-NeRF 360 Bicycle has 156 K init points → 830 K cap.
- `T39 testViewIndices` — manually setting can make the benchmark unusable (all indices > N → no holdouts). Let the `--benchmark` flag set it.
- `T64 mcmcOpacityRegWeight` **and** `T65 mcmcScaleRegWeight` — In 1.4.3 beta set to 0.01, which led to mass extinction (460 K → 5 Gaussians in one iteration). Since 1.4.4 pinned at 0.0, but manually increasing can reproduce the issue.
- `T15 opacityResetInterval` — if not 100 000+ (effectively off) and the training is shorter than 10 000 iterations, the reset destroys convergence. `.preview` therefore has it at 100 000 despite `maxIterations = 5 000`.
- `T54/T55 densifyPhase2*` — two-phase densification ended in tests in a 0-Gaussians cascade. Leave both at 0.
- `T74 useMipSplatting` — Q1.5 closed-no-win 2026-05-25, can even worsen PSNR on some outdoor scenes. Default off, opt-in only for research.

If a field is on this list and you want to change it, first back up your current preset (export as JSON) and consider whether you can reproducibly measure the result — otherwise you won't know afterwards whether you brought about an improvement or a worsening.

CHAPTER

Chapter 7 — Built-in Quality Presets

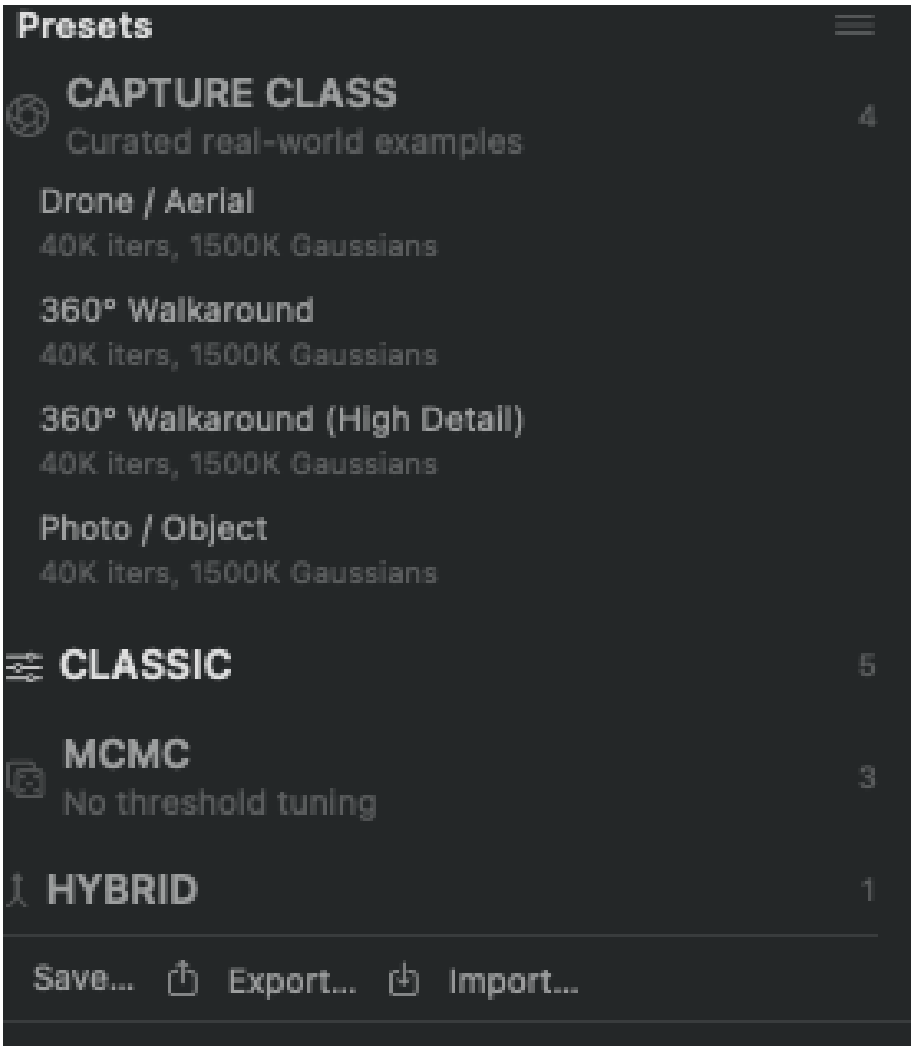

Figure 27: Presets section with all four groups expanded — CAPTURE CLASS (4 presets: Drone/Aerial, 360° Walkaround, 360° Walkaround (High Detail), Photo/Object), CLASSIC (5 presets: Quick/Preview/Balanced/Quality/Ultra Detail), MCMC (3 presets, note „No threshold tuning"), HYBRID (1 preset: Balanced (Hybrid))

WHAT'S IN THE IMAGE Presets section in the Inspector, all four groups expanded. CAPTURE CLASS with the four curated real-world presets (Drone / Aerial, 360° Walkaround, 360° Walkaround (High Detail), Photo / Object) — this is the primary group and the only one visible in Simple mode. CLASSIC with Quick (1K iters), Preview (5K iters, active selection with blue checkmark), Balanced (20K iters), Quality (35K iters) and Ultra Detail (35K iters). MCMC with subtitle „No threshold tuning" — MCMC needs no Densify-Until

threshold: Preview (60K iters, 100K Gaussians), Balanced (120K, 150K), Quality (200K, 150K). HYBRID with the Balanced (Hybrid) (20K iters, 150K Gaussians). Footer action row: Save…, Export…, Import…

A preset is a prepared configuration for training. RadianceKit ships thirteen built-in presets in four groups: four **Capture-Class** presets (P9–P12) — curated recipes for real capture types (drone, 360° walkaround, photo-object), validated by eye against real community material, and the primary axis since v1.6 —, five Classic presets (P1–P5: Quick/Preview/Balanced/Quality/Ultra Detail), three MCMC presets (P6–P8) and one Hybrid preset (P13) that combines the Classic and MCMC strategies. The former „Scene-Class“ presets (Render/3D, Outdoor, Indoor, tuned academically in Phase Q7 against Mip-NeRF-360 and NeRF-Blender scenes) were retired as a visible group in v1.6 — the Capture-Class, validated by eye against real footage, is now the primary axis; the Q7-tuned configurations are kept internally only. You choose the presets in the sidebar under **Presets** or in Simple mode during import. The `+` buttons open dialogs to create your own presets alongside — the thirteen built-in ones cannot be deleted, but can be duplicated.

In Expert view the presets appear grouped by capture type and strategy (Capture Class / Classic / MCMC / Hybrid). A click on an entry writes the stored training configuration into the current state. This is not a snapshot — if you then turn sliders, the state changes, but the preset itself remains unchanged; a colored hint then shows „modified“.

Which preset is the right one when depends mainly on scene type and hardware. The three tabular overviews at the end of the chapter sum this up.

P1 — Quick

WHERE

Inspector → Presets section → Group „Classic“ → Entry „Quick”. UUID suffix `…001`.

TECHNICAL

Diagnostic preset with 1 000 iterations, classic (adaptive) Densification strategy and a training resolution scaling of 0.25× (input image is downscaled to 25 % before training). Not intended to deliver a scene, but to quickly determine whether the setup (camera poses, point cloud, image series) shows any meaningful movement in the loss values at all. On an M3 Ultra typically under 30 seconds on 50–200 images. The small resolution obscures real image quality, but keeps memory footprint and render effort very low. Is also automatically selected as default on first launch when the system has less than 10 GB RAM.

IN PLAIN WORDS

Quick functional test. Images in, wait about half a minute, see if the rough outline of the scene appears. If the image in the viewer looks like a mushy blob — that fits, that’s how it should be. If, on the other hand, you only see dark dots or a totally distorted shape, the camera poses are probably wrong (see Chapter 9). For a shareable result, you then need at least P2 or P3.

P2 — Preview (Classic)

WHERE

Inspector → Presets section → Group „Classic“ → Entry „Preview”. UUID suffix `…002`.

TECHNICAL

5 000 iterations Classic Densification, 0.5× resolution scaling, doubled learning rates compared to standard. Densification (cloning + splitting) is active over the first 2 500 iterations, afterwards only pruning. Default preset for systems with ≥ 10 GB RAM. On an M3 Ultra typically 90 seconds to 3 minutes for a 200-image scene. Delivers a usable impression of geometry and camera pose, but textures are visibly soft — the 0.5× render resolution cannot be directly bypassed afterwards by retraining with P3 or P4, because the learning rates are calibrated to match the half resolution.

IN PLAIN WORDS

The standard for „quickly take a look”. If you've just imported new images and want to see whether the scene can be reconstructed at all, this is the right level. About 2–3 minutes of waiting, then you can rotate in the 3D viewer and judge whether further training runs are worth investing in. Only when the preview result already looks good is Balanced or Quality worthwhile.

P3 — Balanced (Classic)

WHERE

Inspector → Presets section → Group „Classic“ → Entry „Balanced”. UUID suffix `…005`.

TECHNICAL

20 000 iterations Classic Densification at full image resolution. Densification runs over the first 15 000 iterations, from iter 3 000 with a densify interval of 100. Empirically the „sweet spot“ from the documented training sessions: with classic Densification on Horse Full and Truck, L1 loss stabilizes between iter 18 000 and 22 000, longer training brings no significant improvement below Quality (P4). On an M3 Ultra typically 30–60 seconds on 200 images, 5–8 minutes on 1 000+ images.

IN PLAIN WORDS

The „good compromise”. Most scenes already look good with this, without you having to wait an hour. If you want to show the final result somewhere (social media, website, a client demo), it's often enough. Only when you want to zoom into the splat model or you need surface texture details is the jump to P4 Quality or P7 MCMC worthwhile.

P4 — Quality (Classic)

WHERE

Inspector → Presets section → Group „Classic“ → Entry „Quality”. UUID suffix `…003`.

TECHNICAL

35 000 iterations Classic Densification with V546 „Opacity Decay“ (HTGS, Eurographics 2025): after every densify cycle the opacity of all existing Gaussians is multiplied by a factor < 1.0, which reliably removes Gaussians that have become inactive during pruning and thereby achieves 14 % better L1 loss at the same iteration count than the classic 35 000 run. SSIM loss is enabled (`ssimWeight=0.05`). On an M3 Ultra typically 2–4 minutes on 200 images. Achieves final L1 ≈ 0.023 on NeRF-Blender (Lego, Chair, Drums) — best Classic variant in the 560+ documented experiments. Note: needs ~3–5 GB GPU memory; on 8 GB systems P3 is the safe choice.

IN PLAIN WORDS

The best classic variant. Delivers sharp texture and fine geometry, especially on object captures (a sculpture, a chair, a vase). On large outdoor scenes or rooms, however, you barely notice a difference from Balanced — there the switch to an MCMC preset (P6–P8) or a Capture-Class preset (P9–P12) is more worthwhile than the jump from P3 to P4. If you want the absolute maximum of the Classic family, take P5 Ultra Detail.

P5 — Ultra Detail (Classic)

WHERE

Inspector → Presets section → Group „Classic“ → Entry „Ultra Detail”. UUID suffix `…008`.

TECHNICAL

Around 35 000 iterations Classic Densification — the winner of the held-out run of the quality matrix (2026-06-10). On all three tested Mip-NeRF-360 scenes, Ultra Detail beats the built-in MCMC „Quality“ preset (P8) at comparable wall-clock time by about +0.94 dB PSNR. That makes it the strongest quality preset of the Classic group and the sharpest Classic variant RadianceKit ships. On an M3 Ultra typically in the same time frame as P4 Quality (2–5 minutes on 200 images), but needs a bit more GPU memory; on 8 GB systems P3 remains the safe choice.

IN PLAIN WORDS

The sharpest Classic level and the held-out winner of our quality tests: on real scenes about a decibel better than the MCMC „Quality“ variant — at a similar wait. If you want maximum detail fidelity with the proven classic Densification and have enough GPU memory, this is the first choice. If memory is short or you need the smallest possible export file, stick with P4 Quality or an MCMC preset.

P6 — Preview (MCMC)

WHERE

Inspector → Presets section → Group „MCMC“ → Entry „Preview”. UUID suffix `…006`.

TECHNICAL

60 000 iterations MCMC Densification (3DGS-MCMC, NeurIPS 2024) at a cap of 100 000 Gaussians. MCMC replaces the heuristic clone/split logic with Markov chain Monte Carlo relocation: dead Gaussians are relocated via sigmoid-weighted sampling depths, which yields a controlled and reproducible number of Gaussians. The cap hard-caps the maximum count at 100K — this saves memory and render time, but costs detail. On an M3 Ultra typically 5–8 minutes on 200 images. Suitable as an „MCMC functional test“ — helps you judge whether a switch from Classic to MCMC would make sense before you invest more time in P7 or P8.

IN PLAIN WORDS

Like P2 Preview, but with the newer MCMC method. Often delivers more compact, more evenly distributed splat clouds than the Classic variant. For an initial assessment of a scene, the 5–8 minutes are usually enough. If you like the preview result, the next step is P7 (Balanced) or directly P8 (Quality MCMC).

P7 — Balanced (MCMC)

WHERE

Inspector → Presets section → Group „MCMC“ → Entry „Balanced”. UUID suffix `…007`.

TECHNICAL

120 000 iterations MCMC at a cap of 150 000 Gaussians. The middle MCMC level — almost the final Gaussian count of P8 Quality, but only 60 % of the iterations. Empirically the L1 loss in the documented training sessions is at 0.026–0.028 on Horse Full, compared to P8 with 0.0246 — so about 7 % higher, but half the wait time. On an M3 Ultra typically 8–15 minutes on 200 images. Uses a procedure that scales the effective Gaussian cap to the point density of the input SfM point cloud (see T75 in Chapter 6).

IN PLAIN WORDS

MCMC with decent detail depth, but without the long full run of P8. For most scenes this is enough, especially when you want to squeeze an MCMC run into a lunch-break time frame. If memory is tight (e.g. on M processors with only 16 GB), stick here — P8 needs more GPU memory.

P8 — Quality (MCMC)

WHERE

Inspector → Presets section → Group „MCMC“ → Entry „Quality”. UUID suffix `…004`.

TECHNICAL

200 000 iterations MCMC at a cap of 150 000 Gaussians, SSIM loss 0.05, MCMC noise decay over 80 % of the iterations. Best single-run L1 in the 560+ experiments: 0.0238 on Horse Full, averaged over 3 trials 0.0246 (compared to P4 0.0230 on the same scene). MCMC delivers 71 % fewer Gaussians (150K vs ~524K) — crucial when you want to deliver the result on the web, because the smaller cloud produces noticeably smaller export files. Training time on an M3 Ultra typically 20–35 minutes on 200 images; on 1 000+ image sets more like 1–2 hours. Best choice when maximum image quality at minimal final size is desired.

IN PLAIN WORDS

The best MCMC variant. Delivers very clean, compact splat clouds — ideal when you later want to embed the result as a web 3D viewer or send it as a file (the file is smaller than P4 Quality at comparable optical quality). But you need patience — on large captures over an hour of waiting. Plan this rather as an „overnight“ run.

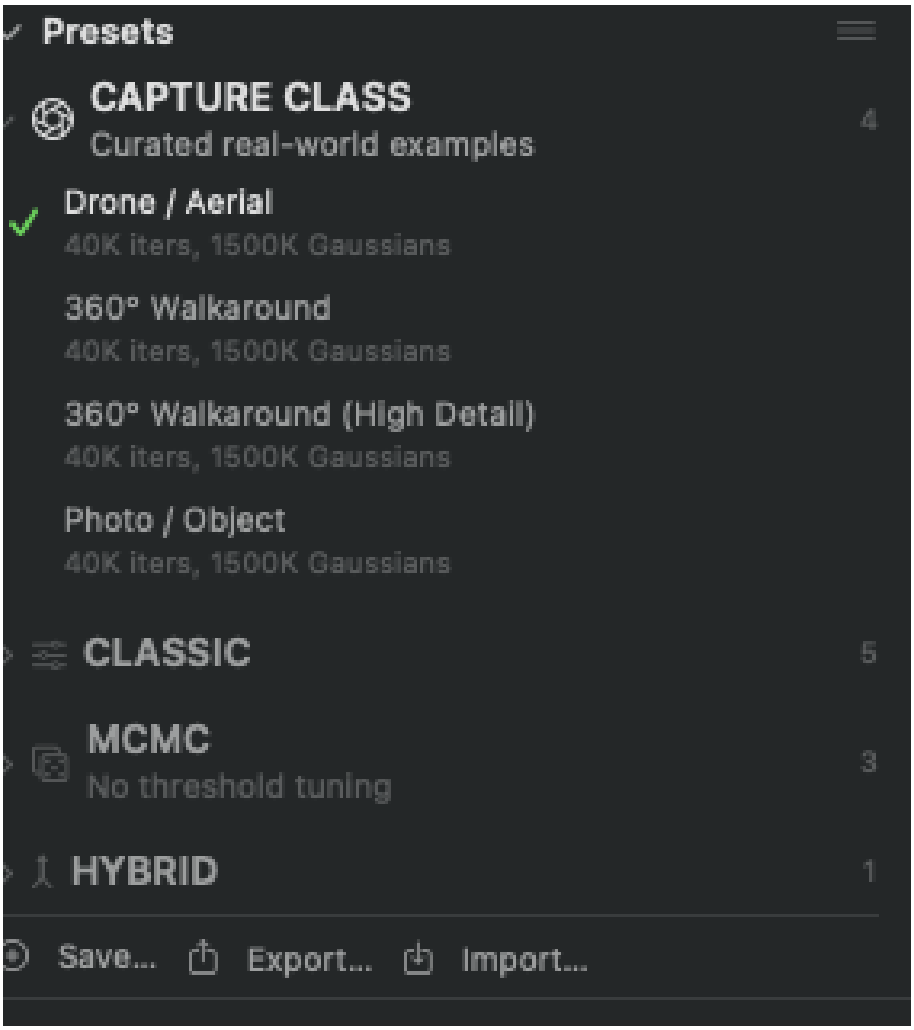

Figure 28: CAPTURE CLASS group expanded with all four curated real-world presets — Drone / Aerial (MCMC, 40K iters, cap 1.5 M), 360° Walkaround (MCMC, 40K, cap 1.5 M), 360° Walkaround (High Detail) (Hybrid, 40K, cap 1.5 M, opt-in) and Photo / Object (Hybrid, 40K, cap 1.5 M). This group sits at the very top and is the only one visible in Simple mode.

WHAT'S IN THE IMAGE Inspector with the CAPTURE CLASS group expanded — the primary preset group since v1.6, the only one shown in Simple mode. Each entry is a recipe for a concrete capture type (drone, 360° walkaround, photo-object), validated by eye against real community material, not a value optimized against an academic test set. Selection by click writes the stored training configuration into the current state.

P9 — Drone / Aerial

WHERE

Inspector → Presets section → Group „Capture Class“ → Entry „Drone / Aerial”. UUID suffix `…010`.

TECHNICAL

Capture-Class preset for aerial and drone orbits of buildings and landscapes. MCMC densifier, 40 000 iterations, cap 1.5 M Gaussians, SSIM loss 0.5 plus edge-aware term 0.1. The decisive part is the anisotropy penalty with weight 0.003 at a ratio threshold of 6 — the „spaghetti killer“ against the typically needle-shaped artifacts that drone footage produces. Validated on a real DJI 4K drone flight over the Pensford viaduct (checked by eye, not just metrically).

IN PLAIN WORDS

For captures from the air — drone flights around a building, over a landscape, along a facade. The strong anisotropy penalty clears away the needle- or spaghetti-shaped artifacts that drone material likes to produce. If your material is captured from the ground, Photo / Object or a Classic preset fits better.

P10 — 360° Walkaround

WHERE

Inspector → Presets section → Group „Capture Class“ → Entry „360° Walkaround”. UUID suffix `…011`.

TECHNICAL

Capture-Class preset for 360° walkaround videos. MCMC densifier, 40 000 iterations, cap 1.5 M Gaussians, SSIM loss 0.5 plus edge-aware term 0.1, gentle anisotropy penalty (weight 0.001 at ratio threshold 15). Person and sky masks are active. The preset expects a 360° equirect video that is internally reprojected to roughly 90°-wide perspective crops before training starts. Validated on 8K 360° walkarounds with a selfie stick (Monument scene, checked by eye).

IN PLAIN WORDS

For 360° walkaround videos — you walk through a room or around an object with a 360° camera or selfie stick. RadianceKit splits the spherical panorama into normal viewpoints itself and masks out passers-by and sky. For maximum sharpness on the same material, additionally try the High-Detail variant (P11).

P11 — 360° Walkaround (High Detail)

WHERE

Inspector → Presets section → Group „Capture Class“ → Entry „360° Walkaround (High Detail)“. UUID suffix `…013` (opt-in).

TECHNICAL

Opt-in Capture-Class preset for 360° walkaround videos with maximum detail. Hybrid densifier (classic abs-gradient clone/split

1. MCMC noise + relocation), 40 000 iterations, cap 1.5 M Gaussians,

anisotropy penalty 0.0015 at ratio threshold 15, SSIM loss 0.2 and edge-aware term 0 — the locked „r50“ screen-split recipe. On 360° material it beats the standard MCMC preset „360° Walkaround“ (P10) at PSNR, LPIPS and visible confetti, and at roughly a third of the splat count. Sits deliberately opt-in *next to* the standard 360 preset until it is validated on more scenes.

IN PLAIN WORDS

The sharper alternative to the standard 360 preset (P10): more detail, less confetti, a noticeably smaller file. Sits deliberately alongside instead of replacing it — so far confirmed on a handful of scenes. If your 360° walkaround is captured cleanly, try this preset first and compare the result with P10.

P12 — Photo / Object

WHERE

Inspector → Presets section → Group „Capture Class“ → Entry „Photo / Object“. UUID suffix `…012`.

TECHNICAL

Capture-Class preset for object orbits from sharp single photos (no video). Hybrid-t1 densifier (with relocation), 40 000 iterations, cap 1.5 M Gaussians, SSIM loss 0.5 plus edge-aware term 0.1, gentle anisotropy penalty (weight 0.001 at ratio threshold 15), opacity decay 0.9995 every 50 iterations, **no** masking. Validated on 163 high-resolution 41-MP photos of a skeleton (checked by eye). Few views (up to about 600) stay below the Hybrid collapse threshold.

IN PLAIN WORDS

For object captures from sharp single photos — you orbit a sculpture, a model, a product with the camera and take photos instead of video. No masking, because sharp photos usually have a clean background. For video sources, take a 360° or Drone preset instead.

P13 — Balanced (Hybrid)

WHERE

Inspector → Presets section → Group „Hybrid“ → Entry „Balanced (Hybrid)”. UUID suffix `…009`.

TECHNICAL

20 000 iterations with the Hybrid densification strategy: classic gradient-driven clone/split places capacity where the loss needs it, MCMC SGLD noise keeps exploring, and dead Gaussians are relocated instead of being lost to pruning. Opacity decay (V546) replaces opacity resets; an anisotropy penalty (weight 0.001, ratio threshold 15) keeps needle-shaped splats in check. The Gaussian cap scales with the scene (150K base, scene-aware ×3.0). Validated on five scenes against pure MCMC at the same budget: on average +0.45 dB PSNR at 20–30 % fewer Gaussians (stonehenge +1.23, family +0.82, garden +0.47 dB). On an M3 Ultra typically 5–10 minutes on 200 images.

IN PLAIN WORDS

A strong first choice for a final result: sharper detail than the MCMC presets at a similarly compact file, in a fraction of the P8 training time. If you only have time for one quality run and none of the capture classes clearly fits, start here. The Classic presets remain better for quick tests, and the Capture-Class presets (P9—P12) are first choice when your scene clearly matches one of those capture types.

Which preset when?

Scenario	First test	Main run
Functional test of new images, < 30s	**P1 Quick**	—
Object orbit from sharp single photos	P2 Preview	**P12 Photo / Object**
Single-object scan (video), < 500 photos	P2 Preview	**P4 Quality** or **P8 Quality MCMC**
360° walkaround video	P6 Preview MCMC	**P10 360° Walkaround** (sharp: **P11 High Detail**)
Aerial / drone orbit, landscape	P6 Preview MCMC	**P9 Drone / Aerial**
Web delivery (small, compact)	P2	**P8 Quality MCMC** (smallest file at full quality)
Sharp detail in little time, compact export	P2 or P6	**P13 Balanced (Hybrid)**
Maximum detail fidelity, Classic strategy	P3 or P6	**P5 Ultra Detail**
Print, marketing, full detail	P3 or P6	**P4 Quality (Classic)** or **P5 Ultra Detail**

Quick comparison

Preset	Strategy	Iters	Max-Gs	Render scale	Typical time (200 images, M3 Ultra)	Q-Sweep
P1 Quick	Classic	1 000	∞	0.25×	~30 s	—
P2 Preview	Classic	5 000	∞	0.5×	2–3 min	—
P3 Balanced	Classic	20 000	∞	1.0×	30–60 s	—
P4 Quality	Classic	35 000	∞	1.0×	2–4 min	V546 HTGS
P5 Ultra Detail	Classic	~35 000	∞	1.0×	2–5 min	Matrix Δ+0.94 dB
P6 Preview MCMC	MCMC	60 000	100 K	1.0×	5–8 min	—
P7 Balanced MCMC	MCMC	120 000	150 K	1.0×	8–15 min	—
P8 Quality MCMC	MCMC	200 000	150 K	1.0×	20–35 min	V544a
P9 Drone / Aerial	MCMC	40 000	1.5 M	1.0×	10–25 min	Eye / viaduct
P10 360° Walkaround	MCMC	40 000	1.5 M	1.0×	10–25 min	Eye / monument
P11 360° Walkaround (High Detail)	Hybrid	40 000	1.5 M	1.0×	10–25 min	Eye (opt-in)
P12 Photo / Object	Hybrid	40 000	1.5 M	1.0×	10–25 min	Eye / skeleton
P13 Balanced (Hybrid)	Hybrid	20 000	150 K	1.0×	5–10 min	Matrix Δ+0.45 dB

Custom presets

Via the **Save…** button in the Presets section (I1 in Chapter 2) you save the current training configuration as your own preset. Custom presets are not „Built-in“ and can be renamed, exported (as JSON), shared via drag-and-drop, duplicated and deleted. The thirteen built-in presets P1–P13 remain untouched by the delete button.

Rule of thumb: If you change something on a preset that you'll want more often — SkyDome on, higher SSIM weight for a specific scene class, different iteration counts — then save the variant as your own preset. That way you know on the next run right away that it's a configuration that deviates from the standard.

CHAPTER

Chapter 8 — Export Formats

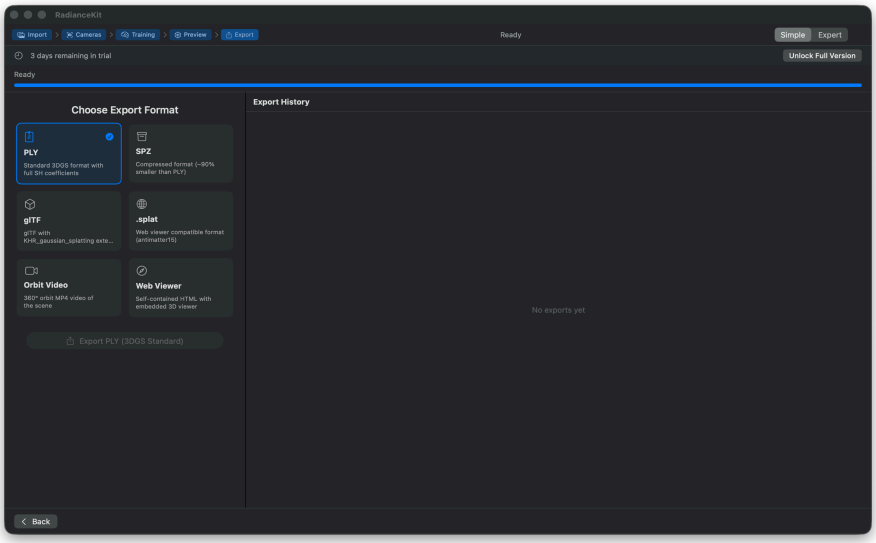

Figure 29: Export format selection in Simple mode — six format cards

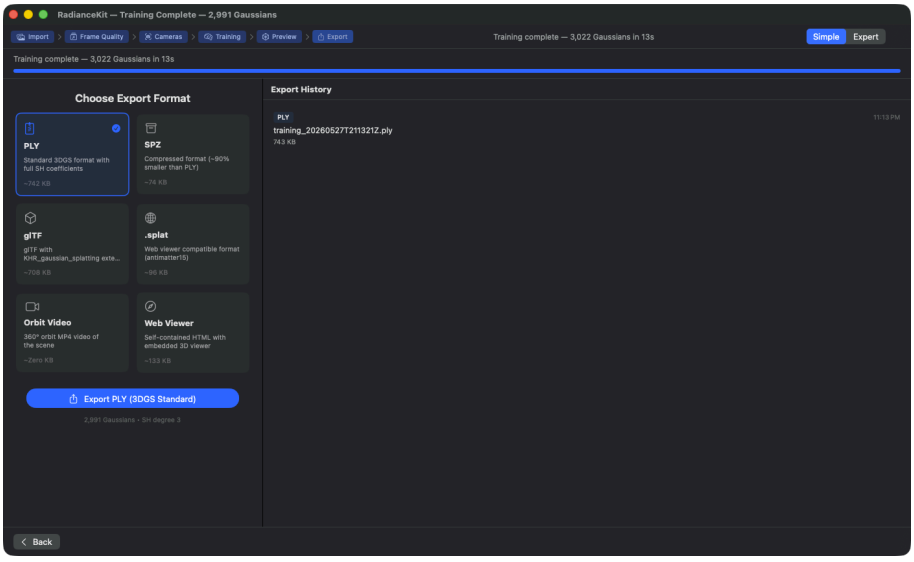

Figure 30: Export format grid live after 5K-iter training on flowers bouquet — all six cards with dynamic size calculation (PLY 742 KB selected, SPZ 74 KB, glTF 708 KB, .splat 96 KB, Orbit Video ~Zero KB, Web Viewer 133 KB), Export History on the right with already saved PLY

What's in the image (2 991 Gaussians, SH degree 3, Bjoern's synthetic Blender bouquet as IP-clean test set): The size figures under each format card are calculated

live from current Gaussian count and format overhead — not hardcoded. From 2 991 Gaussians (SH degree 3) you get 742 KB PLY, 74 KB SPZ (factor ~10× smaller due to quantization), 708 KB glTF (with `KHR_gaussian_splatting` extension, therefore almost PLY-equivalent), 96 KB .splat (compressed 24-byte-per-Gaussian format). Orbit Video shows „~Zero KB", because the size is only known after MP4 encoding. Web Viewer (133 KB) bundles a standalone HTML file with embedded WebGL viewer and compressed splat data — larger than pure .splat due to viewer overhead. Export History on the right lists already completed PLY export („training_20260527T211321Z.ply, 743 KB, 23:13") with format pill and Reveal-in-Finder action.

A completed training delivers a Gaussian cloud — a collection of a few hundred thousand to millions of 3D Gaussian distributions that together reconstruct the scene. RadianceKit knows ten ways to write this cloud to disk. Six of them are pure 3D data formats (PLY, Compressed PLY, SPZ, SOG, glTF, .splat), one bundles the cloud together with a ready HTML viewer (Web Viewer), one renders an MP4 file from an Orbit camera path (Orbit Video), and two export no Gaussian content but only the SfM result (camera poses and rough point cloud) for reuse in other training pipelines (transforms.json + COLMAP Workspace).

Which format is the right one when depends on the goal. For archiving the full data without quality loss you take PLY. For web viewers on your own page, .splat or the built-in Web Viewer is usually enough. If the file has to be minimal, SPZ or SOG is worthwhile. For reusing the SfM result in Nerfstudio, Postshot or Brush, transforms.json and the COLMAP Workspace are the right ways.

All export functions are located in the „Export" menu as well as in Simple mode on the last wizard stage. Most formats are fully sandbox-compliant and work in the App Store version. Only SOG requires an external binary (`cwebp`), which is not necessarily present in the App Store build — details see E4.

E1 — PLY (.ply)

WHERE

Menu bar → Export → 3D Formats → Export PLY… (⌘E). Simple mode: wizard step Export → format card „PLY“. **Size:** typically 100 % (reference value). **Compatible with:** SuperSplat, PolyCam, all 3DGS viewers.

TECHNICAL

PLY is the canonical storage format for 3D Gaussian Splatting. RadianceKit writes a binary little-endian file with the standardized 3DGS property layout: per Gaussian three-component position, three normals always set to zero, three DC SH coefficients (`f_dc_0..2`) for the base RGB color, followed by up to 45 further SH coefficients (`f_rest_0..44`) in the transposed channel-major arrangement defined by the Kerbl 2023 paper (first all R-channel coefficients, then all G, then all B), followed by logit opacity (raw pre-sigmoid values), three log-space scales and a wxyz quaternion rotation. The maximum exported SH degree is clamped to the minimum of user wish and actually learned degree; default is 3 (45 rest coefficients). Before writing, the payload size is calculated in 64-bit integer to catch overflow on extremely large clouds. The file is written atomically, which on large clouds briefly occupies double the disk space.

IN PLAIN WORDS

This is the „original file“. Largest file, highest compatibility, no losses. If you don’t know which format to take, take PLY — that opens in almost every 3DGS tool. For 1 million Gaussians that’s between 200 and 800 MB depending on SH degree. If the file gets too large, look at E2 (compressed PLY) or E3 (SPZ).

E2 — Compressed PLY (.ply)

WHERE

Menu bar → Export → 3D Formats → Export Compressed PLY…. Simple mode: format card „Compressed PLY“. **Size:** approx. 10–20 % vs. PLY (5- to 10-fold compression). **Compatible with:** SuperSplat, PlayCanvas engine, web-based viewers.

TECHNICAL

The PlayCanvas variant of the PLY format with chunked quantization. The Gaussians are grouped in chunks of 256. Per chunk, min/max bounds for position, scale and color are stored separately in the header; the individual Gaussians reference their values relative to these bounds and are compressed to 32 bits each: position and scale with 11-10-11-bit packing, rotation as 2-10-10-10-bit „Smallest-Three“ quaternion, color as 8-8-8-8 RGBA. Higher SH coefficients are quantized with only 8 bits per component (`shCoeffCount * 3` uchar per Gaussian). The format itself is still ASCII-header PLY and therefore in principle validatable with PLY tools, but the vertex properties are declared as `uint` fields. SH degree is by default 0 (no rest coefficients) to maximize compression — higher SH degrees can be selected explicitly.

IN PLAIN WORDS

The space-saving PLY variant. Identical engine compatibility as normal PLY, but 5 to 10 times smaller. SuperSplat and PlayCanvas read it natively. For web deployment almost always better than normal PLY. The quality loss from quantization is usually not visually noticeable, as long as the scene does not contain extremely high-frequency details.

E3 — SPZ (.spz)

WHERE

Menu bar → Export → 3D Formats → Export SPZ…. Simple mode: format card „SPZ“. **Size:** approx. 10 % vs. PLY (90 % smaller). **Compatible with:** Niantic Scaniverse, Niantic Spatial Fields, MetalSplatter.

TECHNICAL

Niantic’s SPZ v2 format. Positions are packed as 24-bit fixed-point (which yields approx. 0.25 mm resolution), scales as 8-bit quantization in log space, rotations as 8-bit Smallest-Three (in v2 only xyz is stored, w is derived in the decoder from the quaternion norm), opacities as sigmoidized 8-bit values. DC SH is stored with an SPZ-specific pack formula (`dc_raw * 0.15 * 255 + 0.5 * 255`), higher SH bands with 5 bits (band 1) or 4 bits (band 2-3) per coefficient. The entire packed binary blob is then compressed with standard gzip (RFC 1952), which yields a gzipped container format with magic bytes `1f 8b`. RadianceKit invokes the system `gzip` for this, because Apple’s built-in zlib API generates proprietary Apple framing that would not be compatible with the SPZ readers in Spatial Fields or MetalSplatter. The system `gzip` also remains spawnable within the macOS sandbox.

IN PLAIN WORDS

The smallest standard file. If you know Scaniverse by Niantic — that’s the format the app uses. Very small, very load-friendly for mobile apps. Directly usable in Niantic’s own cloud viewer (Spatial Fields). About 90 % smaller than a PLY with the same data, while for most scenes optically barely distinguishable.

E4 — SOG (.sog)

WHERE

Menu bar → Export → 3D Formats → Export SOG…. Simple mode: format card „SOG“. **Size:** approx. 5–6 % vs. PLY (15- to 20-fold compression — the smallest option). **Compatible with:** PlayCanvas engine, SuperSplat editor.

TECHNICAL

„Spatially Ordered Gaussians“ — a PlayCanvas format that stores the cloud GPU-ready in several lossless WebP images. First all Gaussians are spatially sorted via 3D Morton code (30-bit Z-order, 10 bits per axis each), which gives the images later cache locality in the renderer. Then positions are quantized to 16-bit values with symmetric log transformation (for better dynamic range) and split into two RGBA images (`means_l.webp` for the lower 8 bits, `means_u.webp` for the upper). Rotations are encoded as Smallest-Three with 3×8-bit plus 2-bit mode in an RGBA image (mode lands in alpha as `252 + largest`). Scales and DC SH are each quantized with a 256-entry codebook (percentile-based distributed over all values), the indices end up in `scales.webp` and `sh0.webp` . The five images plus a `meta.json` with codebooks and bounds are packed into a ZIP file (custom encoder because the sandbox blocks the system `zip`) and saved with the extension `.sog` .

Sandbox warning: SOG is the only format option that requires an external binary. The WebP encoder stage calls `cwebp` from `/usr/local/bin/cwebp` or `/opt/homebrew/bin/cwebp`. If no cwebp binary is found, the code falls back to raw PNG encoding — but: **PNG fallback does not work in SuperSplat**. In the App Store version, evaluate availability based on the build variant; in the developer variant cwebp must be installed via Homebrew (`brew install webp`).

IN PLAIN WORDS

The smallest 3DGS format overall, noticeably smaller than SPZ. But: needs the `cwebp` tool on your Mac, because RadianceKit itself cannot generate all image formats. Install it once with Homebrew, then everything runs. In the App Store version possibly not fully functional — if PNG instead of WebP comes out of the export, you can’t open the file directly in SuperSplat. Anyone working without Homebrew should take SPZ (E3) instead.

E5 — glTF (.glb)

WHERE

Menu bar → Export → 3D Formats → Export glTF…. Simple mode: format card „glTF“. **Size:** comparable to PLY. **Compatible with:** glTF viewers with `KHR_gaussian_splatting` extension (Khronos draft standard).

TECHNICAL

Writes a self-contained `.glb` binary file (no separate bin file attachment) per the `KHR_gaussian_splatting` extension specification. Positions are stored as regular glTF `POSITION` vertex data (float3), all other attributes (rotation as float4, scale as float3, opacity as float, SH coefficients as float3 × shCoeffCount) lie in additional vertex attributes and are referenced via the extension. Important: glTF uses a right-handed Y-up coordinate system, COLMAP/3DGS works Y-down/Z-forward. The exporter therefore applies a 180-degree rotation around the X axis — positions are rewritten as `(x, −y, −z)`, quaternions are adjusted to `(w, x, −y, −z)`. This yields a geometrically correct, handed (not mirror-inverted) display in glTF viewers. JSON and binary chunks are padded to 4-byte alignment as required by the GLB standard.

IN PLAIN WORDS

The official Khronos standard format for 3D data, in the fresh extension for Gaussian Splats. Advantage: glTF is widespread in all major 3D engines (Babylon.js, Three.js, Unity, Unreal). Disadvantage: The extension is still in draft stage in 2026, many viewers can’t read it yet. Sensible mainly if you integrate splat data into an existing glTF pipeline or write a viewer that’s already glTF-capable.

E6 — Splat (.splat)

WHERE

Menu bar → Export → 3D Formats → Export .splat…. Simple mode: format card „.splat“. **Size:** exactly 32 bytes per Gaussian. **Compatible with:** gsplat.js, web-based viewers (antimatter15 reference), most browser 3DGS demos.

TECHNICAL

The antimatter15 `.splat` format — 32 bytes per Gaussian, no header, no indirection. Layout per entry: 3 × float32 position (world coordinates), 3 × float32 scale (exp-transformed from the log space of the internal buffer), 4 × uint8 RGBA color (DC SH coefficient scaled with `SH_C0 = 0.282...` and clamped to [0,255]), 4 × uint8 quaternion (w,x,y,z, normalized and encoded into the byte range as `128 + 128*q`). Only DC SH is stored — higher SH bands are discarded. This makes the format extremely compact, but costs the view-dependent color changes that occur with reflections or specular highlights. The write order is exactly the index order of the cloud (no spatial sorting), web viewers like `gsplat.js` render based on that assumption.

IN PLAIN WORDS

The format of choice if you want to display the splat in your own web viewer with `gsplat.js`. Very compact (32 bytes/Gaussian), but no higher SH degree — so no shiny reflections or subtle color changes depending on viewing angle. For most web applications no problem, because DC color is completely sufficient and the missing view dependence is barely noticeable.

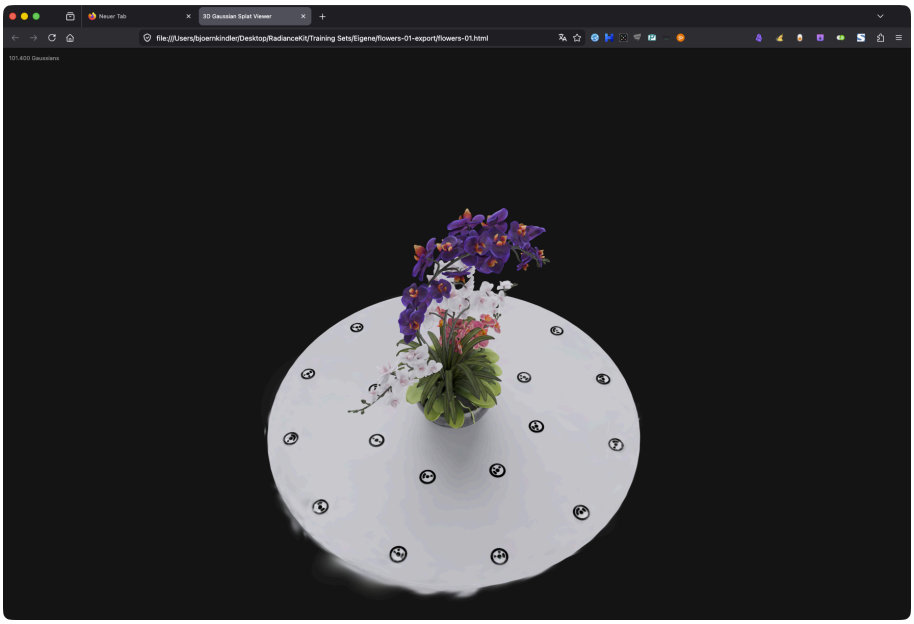

Figure 31: Web Viewer opened in Firefox — Bjoern's bouquet splat rendered with surrounding camera marker spheres, browser tab bar visible on top, no CDN/server setup needed. Standalone `flowers-01.html` opened directly from the Finder by double-click in the default browser — the embedded WebGL2 program renders the Gaussian cloud immediately, without network or server. The black markers around the bouquet are the training cameras, optionally toggleable. Mouse drag rotates, scroll zooms.

E7 — Web Viewer (.html)

WHERE

Menu bar → Export → Media → Export Web Viewer…. Simple mode: format card „Web Viewer“. **Size:** splat data base64-encoded (≈ 4/3 overhead) + approx. 5 KB HTML/JS shell. **Compatible with:** any modern browser with WebGL2 (all desktops, iOS 15+, Android 5+).

TECHNICAL

Bundles the Gaussian cloud together with a fully inline written WebGL2 renderer into a single `.html` file. There are no CDN dependencies, no WASM, no second file. The cloud is internally first encoded as `.splat` binary (same 32-byte logic as E6), then base64-embedded, then decoded with `atob` in the browser. The built-in renderer does its own WebGL2 sorting, mouse orbit controls and CPU sorting per frame; the entire JS code (shader, math, loop) is visible in the output HTML. The axis convention at the storage-to-renderer boundary is exactly the same as in E5: position `(x, −y, −z)`, quaternion `(w, x, −y, −z)`. Optionally a branding overlay can be shown (free-tier toggle). Since everything is inline, the file also works directly from the `file://` protocol — no local web server needed for testing.

IN PLAIN WORDS

A single HTML file that you can send to someone by mail or embed on a website. Double-click in the Finder, and the browser shows your scene with mouse rotation. No upload to a cloud needed, no second file, no server. Ideal for client presentations, portfolio, mail attachments. Disadvantage: the file becomes about a third larger than a pure `.splat` due to base64 encoding — for very large scenes therefore separate hosting of the `.splat` file together with a standard viewer is worthwhile.

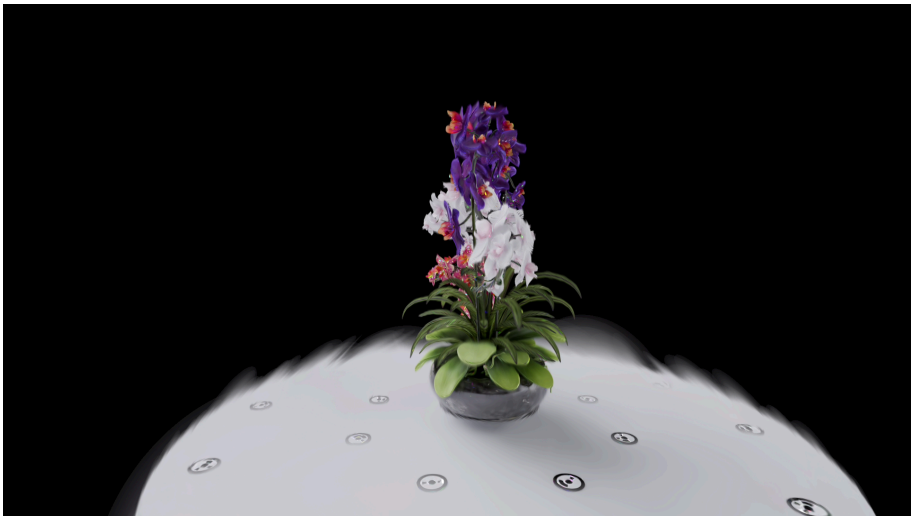

Figure 32: Single frame extracted from flowers-01.mp4 — Bjoern's bouquet in profile render, white platform with camera markers visible, black background (default viewport background, changeable in Settings). The camera orbits the scene on a parametric path (elevation + distance fixed, yaw rotates), duration typically 6–10 seconds at 30 or 60 fps. Frame resolution scalable from 480p to 8K via VideoPreset.

E8 — Orbit Video (.mp4/.mov)

WHERE

Menu bar → Viewport → Record Turntable Video OR Menu bar → Export → Media → Export Orbit Video…. Simple mode: format card „Orbit Video“ with duration slider 3–30 s. **Size:** depends on duration, resolution, bitrate. **Compatible with:** all platforms (H.264 and HEVC are Apple standard).

TECHNICAL

Renders the Gaussian cloud along a parametric Orbit camera path and encodes every frame via AVAssetWriter into an MP4 or MOV file. The Orbit configuration controls rotation speed (revolutions), distance, elevation, FOV, duration and ease-in/out factor. The orbit-video export runs through RadianceKit's OWN ForwardPass with full SH evaluation — pixel-identical to the in-app viewport (WYSIWYG). Per frame the world-alignment matrix (computed by the renderer to rotate the internal coordinates into the Y-up orbit world) is multiplied with the camera, then a camera-convention flip (camFlip: orbit Y-up → COLMAP Y-down) is applied. The offscreen render target is pulled via IOSurface to a CVPixelBuffer for the encoder. The encoder supports H.264 and HEVC, configurable bitrate and resolution from 480p to 8K. Before the first frame the renderer waits 200 ms so that initial splat sorting is completed. This export is GPU-bound — at 8K and millions of Gaussians the render time per frame is several seconds, so total render times of 10–30 minutes for a 6 s video are possible.

IN PLAIN WORDS

A finished MP4 file with a rotation around your scene. Perfect for social media, marketing, presentations. You can set duration (3–30 seconds), rotation direction and speed. The file can be embedded directly on YouTube, Instagram, in PowerPoint and everywhere else. Sometimes slow because the app has to completely render every frame — for an 8K video you can plan five to thirty minutes, depending on the number of Gaussians.

E9 — SfM Transforms (transforms.json)

WHERE

Menu bar → Export → Photogrammetry → Export SfM (transforms.json)…. **Size:** typically 1–10 KB (only poses + intrinsics, no images, no Gaussians). **Compatible with:** nerfstudio, Brush, gsplat, OpenSplat, Meshroom, all modern feed-forward 3DGS trainers.

TECHNICAL

Writes the nerfstudio `transforms.json` format with a list of camera poses plus shared intrinsics. Per camera the view matrix (RadianceKit-internal: World-to-Camera in COLMAP convention) is inverted, after which the camera-local Y and Z basis vectors are mirrored to convert into the nerfstudio convention (OpenGL style, camera looks along `-Z`, `+Y` is up). The final 4×4 matrix lands as row-major nested array of doubles in the `transform_matrix` field of each frame. Intrinsics are stored at the top level (focal length x/y, principal point x/y, image width/height, `camera_model = "OPENCV"`, plus the distortion coefficients `k1, k2, p1, p2`) — except when the exporter detects multiple different intrinsics sets, then they are written per frame. Image paths are written as `images/<filename>` relative to the JSON file; the user must create a sibling `images/` folder with the training photos.

IN PLAIN WORDS

This JSON file describes for every photo where the camera stood and where it looked. The file alone is small and useless — it is used together with the original images in a folder. Nerfstudio, Brush and a few other trainers read exactly this format, and with it you can hand off your RadianceKit SfM results into another tool without the camera reconstruction having to be recomputed there. Saves hours on large scenes.

E10 — COLMAP Workspace (sparse/0/)

WHERE

Menu bar → Export → Photogrammetry → Export SfM (COLMAP Workspace)…. **Size:** three binary files together typically 4–8 MB — `points3D.bin` dominates (one line per 3D point of the sparse cloud), `images.bin` and `cameras.bin` are each noticeably under 100 KB. **Compatible with:** COLMAP itself, Nerfstudio, Postshot, Meshroom, all tools that expect a COLMAP `sparse/` directory.

TECHNICAL

Writes the standard COLMAP `sparse/0/` layout with three binary files: `cameras.bin`, `images.bin`, `points3D.bin`. Format reference is the official COLMAP documentation. `cameras.bin` contains the deduplicated intrinsics list (cameras with identical intrinsics + image size are merged into a single entry); the camera model used is `OPENCV` (model 4), with fx/fy/cx/cy plus the four distortion coefficients k1/k2/p1/p2. `images.bin` lists per image the pose as `wxyz` quaternion plus translation, followed by the camera ID and the filename; no 2D-3D correspondences are stored. `points3D.bin` contains the SfM point cloud with position, color (0-255 RGB) and default values for reprojection and track length. Everything is written in little-endian. Re-import into RadianceKit works via the File menu → „Import COLMAP/Metashape Workspace…“ (see Q3 in the SfM backend chapter).

IN PLAIN WORDS

The official COLMAP format. If you want to continue your training in Postshot, Nerfstudio or another COLMAP-capable software, this is the way. Three small files plus your original images, and the target program accepts it as if COLMAP itself had been the source program. More programs understand this than the transforms.json format (E9), at the same time somewhat less handy because binary instead of text-based.

Which format when?

Goal	Format
Web viewer on your own page	E7 Web Viewer (.html)
Web viewer with `gsplat.js`	E6 Splat (.splat)
Pipeline reuse in Postshot / Nerfstudio	E9 transforms.json + E10 COLMAP Workspace
SuperSplat edit	E1 PLY or E2 Compressed PLY
Niantic Scaniverse / Spatial Fields	E3 SPZ
Maximum compression	E4 SOG (cwebp required)
Marketing/social video	E8 Orbit Video

Quick comparison

Format	Extension	Sandbox	Size (1M Gauss)	Best use
E1 PLY	`.ply`	yes	~250 MB	Archive, highest compatibility
E2 Compressed PLY	`.ply`	yes	~40 MB	Web + SuperSplat
E3 SPZ	`.spz`	yes (gzip spawn)	~40 MB	Niantic + mobile
E4 SOG	`.sog`	conditional (cwebp)	~20 MB	Maximum compression
E5 glTF	`.glb`	yes	~250 MB	Khronos pipeline
E6 Splat	`.splat`	yes	~32 MB	gsplat.js web viewer
E7 Web Viewer	`.html`	yes	~45 MB	Standalone browser file
E8 Orbit Video	`.mp4` / `.mov`	yes	variable	Social/marketing
E9 SfM Transforms	`.json`	yes	~5 KB	Pose handoff
E10 COLMAP Workspace	Directory	yes	~4–8 MB	Pose handoff binary

Size column are rough reference values for 1 million Gaussians with SH degree 3. Real values vary depending on the compressibility of the scene; SH degree 0 reduces PLY/glTF by a factor of 4.

CHAPTER

Chapter 9 — SfM Backends

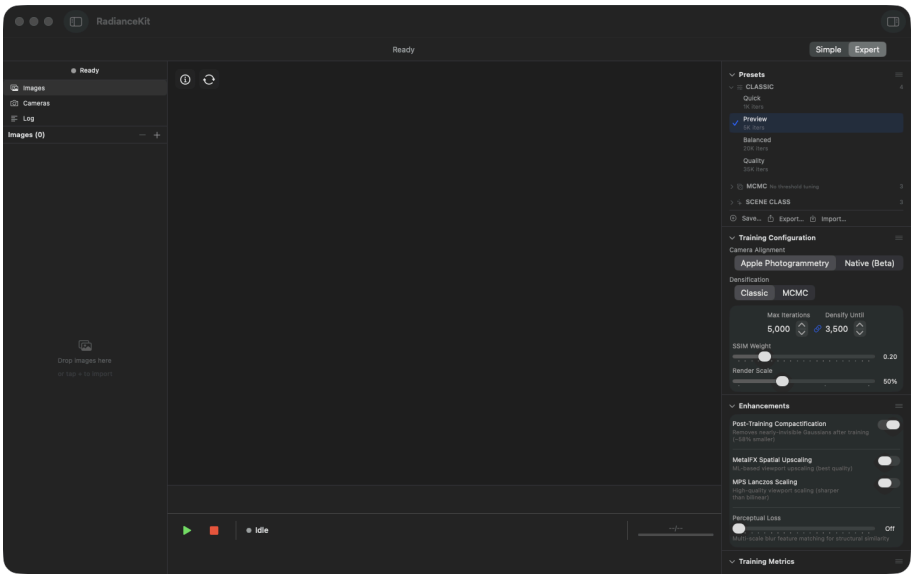

Figure 33: Expert Mode with Camera Alignment picker in the Inspector (Apple Photogrammetry / Native (Beta))

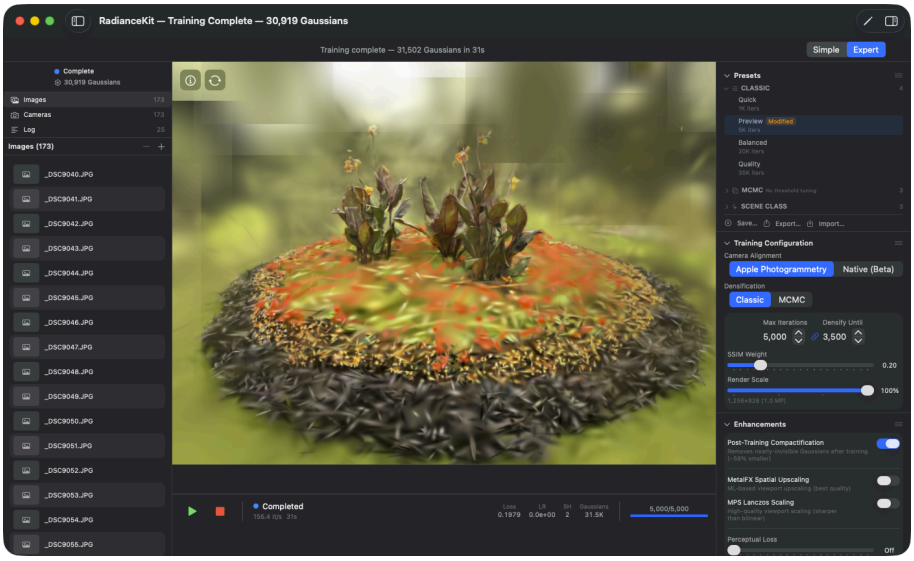

Figure 34: Inspector with Native (Beta) active — Camera Alignment picker second option selected, all other training configuration parameters unchanged

WHAT YOU SEE IN THE IMAGE The Camera Alignment picker in the Inspector is a segmented control with two options — Apple Photogrammetry (default for App Store builds, fully sandbox-compliant) and Native (Beta) (RadianceKit's own FAST+BRIEF+GLOMAP

pipeline backend, developed during Phase 3.8/3.9, as of 2026-05). Native (Beta) is orbit-only validated and faster than Apple Photogrammetry with ≥1 000 frames, but does not yet meet the Phase-3 §5 quality gate (finalLoss ≤ 0.0115) — hence the Beta tag. External SfM results from Metashape, COLMAP or any other photogrammetry software can additionally be imported via the File menu (Q3 COLMAP text format, Q6 Workspace Import) — the picker does not switch, but the imported poses replace the SfM result.

SfM stands for **Structure from Motion**. From a set of overlapping photos, the software reconstructs for each image the position and view direction of the camera in a shared 3D coordinate system. In the process, a coarse 3D point cloud is generated, which initializes training with Gaussian Splatting. The SfM result is the input for the actual training and decisively determines the later image quality.

RadianceKit offers five SfM paths: two backends built into the app (Q1 Apple Photogrammetry, Q4/Q5 Native), two import paths from external tools (Q3 COLMAP text format, Q6 binary Workspace Import) as well as Q2 COLMAP binary, which is only available in developer builds outside the App Store. Which is the right one depends on the scene type (orbit around an object, indoor space, drone flight) and on whether external software already provides a reconstruction.

Q1 — Apple Photogrammetry

WHERE

Expert View → Inspector → Training Configuration → Camera Alignment picker, entry “Apple Photogrammetry”.

TECHNICAL

Wraps Apple’s built-in photogrammetry framework, originally developed for Object Capture. Apple internally extracts features with a proprietary pipeline (steps are not publicly documented), verifies them via multi-view matching, and solves bundle adjustment on the Apple Silicon Neural Engine + GPU. The backend is fully App Store compliant (no external binary, Sandbox=true, on-device), but only delivers camera poses plus a coarse point cloud — no diagnostic metrics like track length or reprojection error. Scales according to Apple’s recommendation up to a few hundred images. With more than ~500 frames in linear drone flights or large outdoor scenes, reproducible crashes or silent dropping of individual cameras have been observed.

IN PLAIN WORDS

This is the simplest way. Put images in, the app calculates. Works very well for classic object scans — when you walk around a piece of furniture or a sculpture and take 50–200 photos. For drone flights over landscapes or with very many images (over 500), Apple’s method tends to become unstable, however. For such scenes test the Native backend (Q4/Q5) or compute the cameras in Metashape and load them via Workspace Import (Q6).

POWER-USER

Q2 COLMAP binary — spawns the external COLMAP program as a subprocess and is therefore **not available** in the App Store version (sandbox). Only works in developer builds outside the App Store. For the quality that COLMAP delivers, there is Workspace Import in the App Store version (Q3 or Q6): compute the SfM in COLMAP or Metashape externally and load the result.

Q3 — COLMAP text format (Metashape / ETH3D)

WHERE

Menu “File → Import COLMAP / Metashape Workspace…” (Cmd+⇧+I) OR drag-and-drop of a folder with `sparse/0/cameras.txt`.

TECHNICAL

Reads the standardized COLMAP text export — three text files `cameras.txt`, `images.txt`, `points3D.txt` in the `sparse/0/` subfolder — and converts to the internal SfM result model. Same format definition as the COLMAP binary export, just as ASCII instead of binary. Exported by Agisoft Metashape, RealityCapture, PolyCam and the ETH3D benchmark in exactly this layout. The parser shares camera model detection with the binary parser (all common models: SIMPLE_PINHOLE, PINHOLE, OPENCV, OPENCV_FISHEYE, FULL_OPENCV). Robust against comment lines and empty lines. Scales in tests up to ~1 400 cameras (ETH3D Tunnel) without issues.

IN PLAIN WORDS

If you have already worked with Metashape, RealityCapture or another commercial photo-3D software and exported the result — you can load this export directly in RadianceKit, without the app having to recompute. This saves hours of waiting time. Just load the entire folder via the File menu or drag it into the window.

Q4 — Native SfM (incremental)

WHERE

Expert View → Inspector → Training Configuration → Camera Alignment picker, entry “Native (Beta)”. Incremental is the default mode of this backend — there is no separate mapper picker in the Inspector. Via CLI the mode can be set explicitly with `--native-sfm` or `--sfm-mapper incremental`.

TECHNICAL

Own GPU-accelerated implementation of the entire SfM pipeline: FAST+BRIEF features OR SuperPoint+LightGlue via CoreML (with `--coreml-features`), followed by Hamming-KNN matching, RANSAC fundamental matrix, track building, initial pair selection, two-view bootstrap (F→E plus DLT), greedy incremental mapper with PnP registration and multi-view triangulation, and final bundle adjustment via Schur-reduced Levenberg-Marquardt with Huber loss and analytical Jacobians via Cholesky solve. Fully App Store compliant: no external binary, Sandbox=true. With the R2 collapse detector shipped in Phase 3.10: if the app registers fewer than 60 % of the input frames or the points-per-camera rate falls below 13, it automatically falls back to the global mapper (Q5). Empirically clean on orbit/turntable scenes; on more general motions (drone flight, indoor spaces with complex geometry) the success rate is lower — the detector catches these cases, though. Scales reliably up to ~200 cameras, higher with significantly longer runtime.

IN PLAIN WORDS

Apple’s strengths (App Store compatible, fast for orbits) with additional diagnostic values. Works particularly well when you walk around a subject as for an Object Capture. With more complicated recordings (drone flight or living room), RadianceKit automatically detects that this won’t work and switches to the global method. Marked “Beta” because still being tested — the standard recommendation remains Apple Photogrammetry for simple object scans and Workspace Import (Q3 or Q6) for demanding outdoor sets.

Q5 — Native SfM (global)

WHERE

Is called automatically when the incremental mapper (Q4) triggers the collapse detector (fewer than 60 % of input frames registered or points-per-camera rate below 13). Manually forceable only via CLI `--sfm-mapper global`. In the Inspector the global method is not reachable via a separate picker — the app decides on its own when to switch.

TECHNICAL

Global variant of the native pipeline. First feature extraction + matching as in Q4, then relative pose estimation for all verified pairs, followed by rotation averaging (synchronizes all camera rotations in the world coordinate system) and translation averaging (LSQR-based on a matrix-free sparse formulation to avoid integer overflow with large camera counts). Scales to ~5 000 cameras in principle, in practice quality-degraded above a few hundred cameras — the Phase 3.8 §5 acceptance gate measurement on K-1351 gave finalLoss 0.07 instead of the targeted 0.0115. Treated as a “fallback tier”: comes into play when the incremental mapper degenerates, but is not itself re-checked for quality.

IN PLAIN WORDS

The plan-B path for the native engine. Is called automatically when the faster incremental path fails. Delivers a usable result, but with very large or difficult scenes is usually not as precise as what you get from Metashape or an external COLMAP installation. If Native becomes your standard workflow, the detour via Workspace Import (Q3 or Q6) is worth it in such cases.

Q6 — Metashape / COLMAP text workspace import (Phase Q7)

WHERE

File menu → “Import COLMAP / Metashape Workspace…” (Cmd+⇧+I). Drag-and-drop of a folder with `sparse/0/cameras.{bin,txt}` and `images/`.

TECHNICAL

Automatically detects whether a folder selected via drag-and-drop or open panel matches one of the three COLMAP workspace layouts (`sparse/0/`, `sparse/`, or root) and whether the reconstruction is binary (`cameras.bin`) or text (`cameras.txt`). The binary path uses the COLMAP binary parser, the text path the ETH3D loader — both produce the same SfM result model and the rest of the pipeline (import images, start MCMC training) is agnostic to the source. The images are opened via the app sandbox bookmark system security-scoped, so the import also works in the App Store version. Specifically intended for the case “Metashape export without recomputing reconstruction”. The detection mentioned in the File menu entry warns in the app log if the chosen folder is not a recognizable workspace.

IN PLAIN WORDS

Specifically the Metashape user function. If you have a license for Metashape or RealityCapture and did the camera reconstruction there, you can simply drag the export folder in here and immediately start training. Saves several hours of compute time on large scenes, because RadianceKit then does not do the SfM itself.

Which backend when?

Scenario	Recommended backend
Object scan, 50–200 photos	Q1 Apple Photogrammetry
Large outdoor / drone / >500 images	Q6 Workspace Import (compute in Metashape or COLMAP, then load)
Metashape/RealityCapture export available	Q6 Import (no SfM needed)
ETH3D / academic COLMAP text set	Q3 COLMAP text import
Strictly App Store compliant + orbit scene	Q4 Native incremental
Q4 fails	Q5 Native global (automatic)
ETH3D benchmark data	Q3 (autotest precomputed)

Quick comparison

Back-end	App Store	Sand-box	External binary	Best Use	Max ~Cams
Q1 Apple PG	✅	✅	—	Orbit-Ob-ject	~300
Q2 COLMAP Binary	❌ (devel-oper build only)	—	colmap/glomap	Outdoor large	~5 000
Q3 COLMAP text im-port	✅	✅	—	Bench rigs	~1 500
Q4 Na-tive in-cremen-tal	✅	✅	—	Orbit-Ob-ject	~200
Q5 Na-tive global	✅	✅	—	Q4 fall-back	~1 351
Q6 Work-space Import	✅	✅	—	Metashape reuse	per source

CHAPTER

Chapter 10 — Simple Mode

Simple Mode (German Einsteiger-Modus, Cmd+1) is the guided workflow for everyone reconstructing a 3D Gaussian Splatting scene for the first time. Instead of showing a sidebar full of Inspector fields, the app guides you through four steps: first import images or a video and choose a Quality Preset, then processing runs (SfM + Training), afterwards the finished scene can be inspected in a 3D preview, and finally it is exported to the desired format. A slim progress bar at the top edge of the window always shows which step you are currently on.

Compared to Expert Mode (Cmd+2), which shows all control panels simultaneously, Simple Mode hides unused options, gives validation warnings for too few or poor images, and on each step only offers the buttons that make sense in the current state. You can switch between Simple and Expert Mode at any time (Cmd+1 / Cmd+2), the entire state — imported images, chosen preset, currently running training, finished point cloud — is preserved and is immediately available in the other mode.

Z1 — Import (choose images & preset)

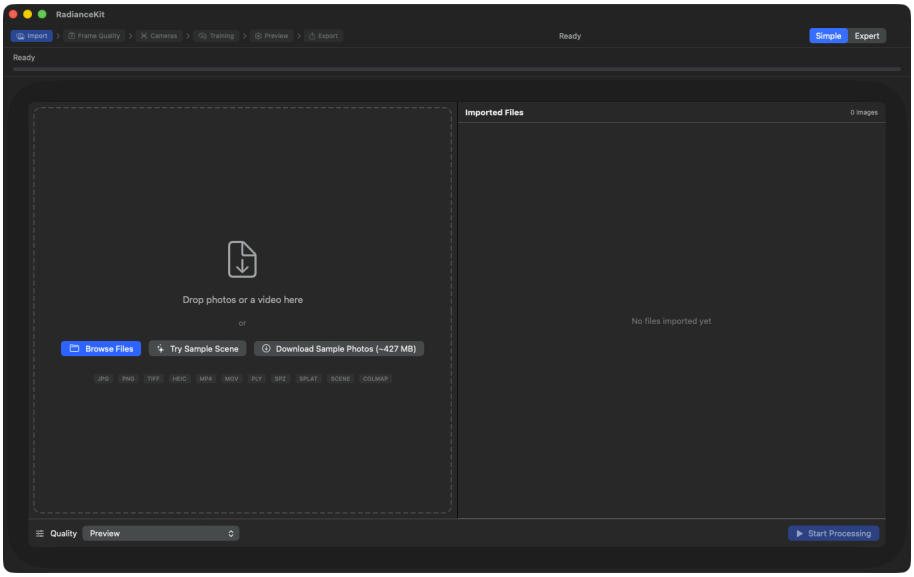

Figure 35: Simple Mode step 1 — empty drop zone before image import, crumb trail at the top (Import → Frame Quality → Cameras → Training → Preview → Export), format pills JPG/PNG/TIFF/HEIC/MP4/MOV/PLY/SPZ/SPLAT/SCENE/COLMAP

WHAT YOU SEE IN THE IMAGE Crumb trail (Import active) shows the four-step workflow. Left drop zone with three CTAs: “Browse Files” (NSOpenPanel), “Try Sample

Scene" (bundled demo), "Download Sample Photos (~427 MB)" (Mip-NeRF360 flowers subset). Format pills underneath list all accepted file types. On the right "Imported Files" with counter "0 images" and empty state "No files imported yet". Below, Quality picker (default: Preview) and "Start Processing" (disabled as long as no images are present).

The first step consists of giving the app image material. Via drag-and-drop into the large dashed field in the middle, via the "Browse Files" button, or by clicking on the bundled sample scene. On the right a list of all imported images appears with resolution and file size; below in the floating toolbar you choose the Quality Preset and start the pipeline with "Start Processing". Validation warnings (red at < 3 or < 10 images, orange at 10–19) indicate whether the app expects a sensible reconstruction or not.

C-01 ProgressIndicator (step indicator)

WHERE

Top above the workflow, always visible.

TECHNICAL

Shows a horizontal progress bar across the entire pipeline (Frame Quality → SfM → Training) with stage allocation: Frame Quality occupies 0–5 % (Phase 3.11, very short), SfM occupies 0–30 % of the bar, Training 30–100 %. Next to it status text and phase-named percent display ("SfM 41 %", "Training 12 500/20 000"), so users do not read the apparent regression "41 % SfM → 25 % Training" as an error — the bar shows the overall pipeline progress, not the sub-stage. ETA calculation starts as soon as enough training speed is measured (typically after the first 100 iterations). The same display is also used in Expert Mode above the Inspector.

IN PLAIN WORDS

The slim bar at the very top is your map through the workflow. It tells you not only what the app is currently doing (aligning cameras, training running, …), but also how far it has progressed overall. The split is intentionally such that the camera computation occupies the first third of the bar and the actual training the rear two thirds — otherwise it would seem as if progress had suddenly jumped back to zero after SfM. So you can lean back relaxed, a glance at the bar is enough to see the rough stage. The text next to it tells you whether you are currently in the SfM stage (e.g., "SfM 41 %") or in training (e.g., "Training 12 500/20 000"), so the numbers do not seem confusing. If the ETA is not shown, training is simply still too young — the app only estimates once it has measured enough speed.

C-03 DropZoneView (drag-and-drop area)

WHERE

Left side of the import step, large dashed rectangle with symbol. Displayed in Simple Mode with the label “Drop photos or a video here”.

TECHNICAL

Drop area that briefly bounces the symbol and tints the background as soon as drag items hover over the field. Accepts JPG, PNG, TIFF, HEIC, MP4, MOV, PLY, SPZ, .splat, .radiancescene bundles, and directories. Drop routing by type: images are collected and passed in sorted order, videos trigger the frame-sampling path, splat files open the preview directly, scene bundles are read in. Directories are enumerated and all contained images are imported. Security-scoped bookmarks for sandbox-compliant access are correctly acquired and released. Unsupported extensions are displayed as a warning banner for 5 seconds.

IN PLAIN WORDS

The large dashed field is the main control of the first step. Just drag photos or a video into it, or a whole folder — the app takes everything it knows, and ignores the rest. When the field turns blue and the symbol briefly bounces, the app has recognized the drag. Release, and the import starts immediately: images move into the list on the right, videos automatically trigger the frame-sampling step, and already-trained `.ply` / `.spz` / `.splat` files open the preview directly. If a format does not fit at all (e.g., PDF or BMP), a brief notice appears at the top edge — the app does not silently swallow unknown material.

C-05 Browse Files Button

WHERE

Inside the drop zone, prominent button.

TECHNICAL

Button that opens the macOS file dialog with multiple selection and the file types JPG, PNG, TIFF, MP4, MOV, folders as well as the app-own scene format. Result URLs are security-scoped and are routed through the same import paths as drag-and-drop. When the user selects a folder, it is recursively enumerated for images.

IN PLAIN WORDS

If drag-and-drop is inconvenient for you, just click this button and navigate in the macOS file dialog to your photos. You can select multiple files at once (Cmd-click on the individual images) or select an entire folder — the app then recursively searches the folder for all supported image types. This is particularly practical when your recordings are nested in subfolders (e.g., “shoot-day1/”, “shoot-day2/”) — one click on the main folder is enough. Functionally the button does exactly what drag-and-drop also does; just choose the path that is more convenient for you.

C-06 Try Sample Scene Button

WHERE

Inside the drop zone, only visible when the app bundle contains the sample scene and no images/splats have been imported yet.

TECHNICAL

Only appears when (a) a `sample-scene.splat`, `.spz` or `.ply` is present in the app bundle AND (b) no images/videos have been imported and no point cloud is present. On click loads the finished point cloud (preferring the smallest format — `.splat` ~3 MB, `.spz` ~1.4 MB, fallback `.ply`) and after 400 ms sets hardcoded camera values from the original metadata of the flower scene for an aesthetically pleasing entry perspective.

IN PLAIN WORDS

If you launch the app for the first time and just want to see what comes out at the end — click here. That opens a finished trained flower scene that you can immediately rotate and export, without the app having to compute. The camera is preset to an aesthetically pleasing entry perspective, so you immediately see something beautiful. Perfect for trying the 3D controls and the export step risk-free, before you tackle your own recordings. As soon as you import your own images, the button disappears automatically — it is only shown as long as the project is completely empty.

C-07 Download Sample Photos Button

WHERE

Inside the drop zone, next to “Try Sample Scene”; same visibility conditions.

TECHNICAL

Triggers a download (repo `github.com/bkindler/radiancekit-sample-photos`) that loads approx. 427 MB of 960 full-resolution frames and feeds them into the app. During the download the button is disabled. The progress appears in the top progress bar as “Downloading X %” in its own stage, because this stage keeps its own 0–100 % scale and does not overlap the later SfM stage.

IN PLAIN WORDS

Just like the sample scene, only with the source photos instead of the finished result. This way you can let the entire pipeline run once yourself and see how long SfM and training really take on your Mac. The download is large (about half a DVD = 427 MB), but only happens once — afterwards the photos are local and you can restart the pipeline as often as you want with different presets. While the download is running, the top progress bar shows the current download status in percent, so you can estimate when it will start. Tip: use a fast Wi-Fi or wired network — the 427 MB will otherwise take a while.

C-09 Quality Presets Picker

WHERE

Floating bottom toolbar of the import overlay, to the left of the Start button.

TECHNICAL

Control with label “Quality” groups the available presets by category (Classic / MCMC / Custom). Built-in presets are grouped by category; the section headers are hardcoded. Custom presets only visible when some exist. Locked state: presets not in the free list (Quick + Preview) get a “🔒” suffix at the name when the user has not purchased; on selection the picker jumps back to Preview and automatically opens the purchase sheet. On selection the preset is applied, which replaces the entire training configuration.

IN PLAIN WORDS

Here you choose how accurately and how long the app should compute. “Quick” and “Preview” can be used without purchase and deliver an initial result in a few minutes — ideal for testing whether your images are sensible at all. “Balanced” and “Quality” need the full version and deliver significantly cleaner models, but take hours instead of minutes. MCMC is a different strategy that gets by with fewer Gaussian splats — good if you want to export the model compactly later or put it on the web. You recognize premium presets by the small lock symbol at the name; if you tap one without a license, the picker jumps back to Preview and the purchase sheet opens automatically. Rule of thumb: always start with Preview, look at the result, and then decide whether a longer run is worth it.

C-10 Start Processing Button

WHERE

Floating bottom toolbar of the import overlay, to the right of the preset picker.

TECHNICAL

Button that stays gray as long as neither images nor a video have been imported. On click starts the pipeline and switches the stage machine in the order Frame Quality → SfM → Training. The button itself has no further status; a running processing instead appears as a separate processing screen.

IN PLAIN WORDS

The “Go” button. As long as it is gray, input images or a video are still missing. Once you have dragged in photos, it becomes active and you click it to start SfM and training in sequence. From there the app takes over the entire workflow and you automatically land on the processing screen (Z2). You don’t have to click anything else — only after training ends does the app switch back to the preview (Z3). If you change your mind, you can also still cancel anytime via Cancel afterwards.

C-11 Video Sampling Slider

WHERE

Right image list, visible only when a video (instead of images) has been imported.

TECHNICAL

Slider 0.5 fps – 30 fps in 0.5 steps. On change the frame density is updated and additionally the number of target frames (at least 10) is computed from density and video length. The slider sits outside the image list, because list items would block mouse events from sliders. Below the slider the computed target frames ("247 frames") and the video length ("1m23s video") are shown. Tooltip warns: "Doubling the density doubles the number of frames and increases SfM time by ~100%."

IN PLAIN WORDS

If you imported a video instead of photos, this slider decides how many individual images the app should extract from the video. More images = better quality, but linearly more compute time. For a 30-second orbit video, 5 fps (150 images) is a good start; for 1-minute recordings, 3 fps is often completely sufficient. Under the slider the app shows live how many frames result at the current setting — so you immediately see whether you hit the sensible range of about 100–300 images. If the result becomes poor, pull the slider to the right and try again; but doubling the frame rate also roughly doubles the SfM duration.

C-12 Clear All Button

WHERE

Right image list, bottom right; visible only when images have been imported.

TECHNICAL

Red button. Click opens a confirmation dialog with title "Clear all imported files?" and message "N images will be removed.". Confirmation clears all imported images/videos, staging directories, the point cloud, training status, the SfM result, and all caches; the stage jumps back to Import. On Cancel everything is preserved. The dialog is configured as a non-destructive default path (destructive button marked red).

IN PLAIN WORDS

If you want to start completely over, click here. The confirmation prompt appears because deletion discards all current imports including any already-computed cameras and training results — you cannot undo it. Useful when you want to completely replace the chosen image material or get rid of an old project before starting a new one. Note: removing a single image is done via the list on the right (see next item), not via this button. Your files on disk are not deleted in the process — the app only forgets its references.

C-13 File List ForEach (single image removal)

WHERE

Right image list, every entry.

TECHNICAL

List over the imported images with swipe-to-delete. Per image one row with icon, file name, resolution ("1920 × 1080") and file size (formatted KB/MB). Resolution comes from a metadata cache populated asynchronously from the image headers, so the UI does not block. The delete action offers macOS-typical swipe delete (trackpad swipe left on a row) as well as keyboard delete for the selected row. Note: the extended image-delete path with explicit minus button, backspace, and Cmd-Z for undo was added *only in Expert Mode* in the Project Navigator — in Simple Mode it stays with swipe delete.

IN PLAIN WORDS

The list on the right shows each imported image with resolution and file size — practical for seeing at a glance whether you have mixed high-resolution with low-resolution material together. To remove a single image, swipe it with two fingers to the left on the trackpad — as in iOS Mail — or select it and press Delete. The app does not delete the file itself; it only takes it out of the current project. If you need a proper minus button or Cmd-Z undo, switch to Expert Mode (Cmd+2), there it exists in the Project Navigator. In Simple Mode it deliberately stays with the simple swipe pattern.

C-15 Validation Warnings (3-tier)

WHERE

Below the image list, above the Clear All button.

TECHNICAL

Three successive thresholds based on the number of imported images (only active when images present and no video): - < 3 images: red banner (red octagon), text "At least 3 images are required. Camera alignment cannot be computed from fewer images." - 3–9 images: red banner, text "With fewer than 10 images, SfM often fails and the trained scene tends to overfit [...]. 15–20 images minimum recommended; 30+ for object captures." - 10–19 images: orange banner (warning triangle), text "Workable, but quality usually improves with 20+ images and good coverage around the scene."

From 20 images the banner disappears. Thresholds are hardcoded and based on empirical 560+ training experiments.

IN PLAIN WORDS

The app looks at how many images you have imported, and gives you a color-coded assessment. Red means: this will most likely fail — either SfM cannot compute cameras or training overfits on too little material. Orange means: might work, but don't expect top quality, because the algorithm finds little overlap between the images. No banner means: good conditions, you have enough material. If you really want clean models, aim for at least 30–50 evenly distributed shots around your subject — happily also significantly more for outdoor scenes or large rooms. You can start despite a warning, but don't be surprised if SfM aborts without comment or the model looks holey.

C-16 COLMAP Workspace Detection

WHERE

On dropping a folder — not a visible button, but detection logic.

TECHNICAL

On dropping a directory it is checked whether it contains one of the three canonical workspace layouts: `sparse/0/cameras.bin`, `sparse/cameras.bin` or directly `cameras.bin` in the root. If so, the standard image enumeration is aborted and instead a modal alert is opened asking the user whether the existing reconstruction should be used or the images should be re-run through Apple Photogrammetry. Same path also for text-format workspaces (cameras.txt) and ETH3D exports. See Chapter 9 backend Q6 for details. Works in Simple Mode the same as in Expert Mode.

IN PLAIN WORDS

If you have already worked with Metashape, RealityCapture or COLMAP and ran the camera computation there, you can simply drag the export folder in here. RadianceKit detects automatically from the content that it is a COLMAP workspace (it checks for `sparse/0/`, `cameras.bin` and the like), and asks you whether it should take over the finished computation or compute it itself. Taking over saves hours of waiting time on large scenes, because SfM is completely skipped — the training starts immediately. Text-format workspaces (cameras.txt) and ETH3D exports are also detected. This function is available in Simple Mode the same as in Expert Mode; more details are in Chapter 9 under backend Q6.

When to move to the next stage?

You can click Start Processing as soon as (a) at least one image or a video has been imported and (b) the validation banner is orange or gone. With a red banner the app does let you start, but you can with high probability cancel the processing again right away. Recommended: at least 20 images, sharp, with clear overlap between successive shots, all from approximately the same distance to the subject. Choose a preset before starting that fits your time budget — with 30 images and the Quick preset you are done in a few minutes, with Quality it rather takes 1–2 hours.

Z2 — Processing (SfM + Training)

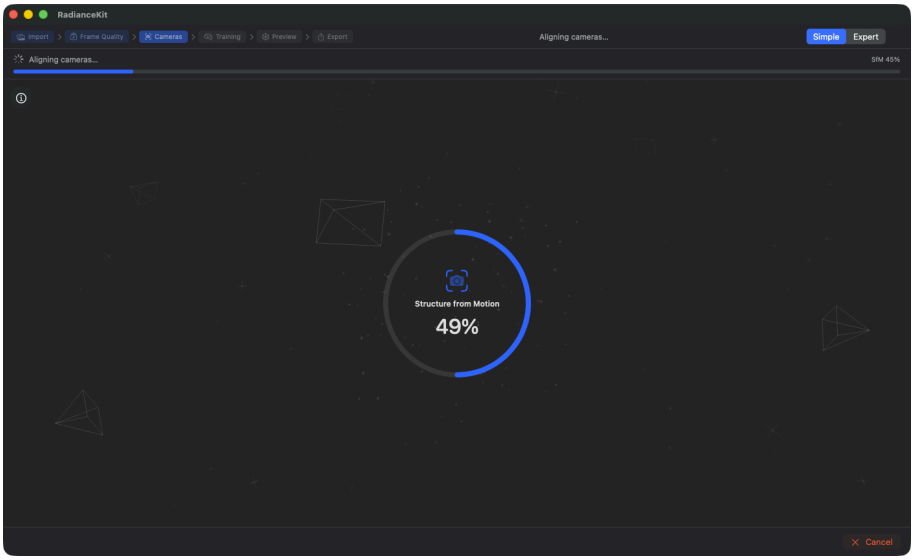

Figure 36: Z2 SfM phase — stage icon "Structure from Motion" with 41 % in the large circle, top status bar at "SfM 25 %", Cancel button bottom right

SfM phase (cameras are being aligned): Large progress circle shows sub-stage progress (here 41 % of the running Apple Photogrammetry session). Status text "Aligning cameras…" top left. Crumb trail marks "Cameras" as the active stage. Top status bar shows pipeline overall progress (25 %) — SfM occupies the first half of the bar. Floating wireframe cameras in the background hint that poses are being estimated.

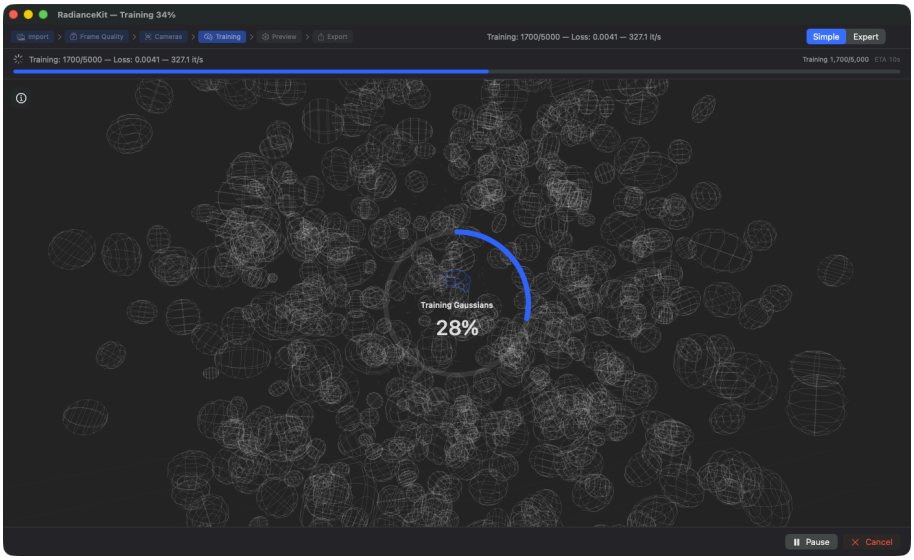

Figure 37: Z2 training phase — stage icon "Training Gaussians" at 6 %, live metrics on top (Training: 400/5000 — Loss: 0.1642 — 138.7 it/s), ETA 33s, Pause/Cancel at the bottom

Training phase (Gaussians are being optimized): Sub-stage icon switches to "Training Gaussians", percent counts iterations of the chosen preset (here 400 / 5 000 for Preview preset = 8 % of the stage). Live metric line shows loss value (0.1642), iterations per second (138.7 it/s) and ETA (33 s). Pipeline overall progress climbs from 50 % to

100 % during this phase. Pause button (instead of cancel-only in SfM phase) allows resume later; Cancel discards the training result and returns to Z1.

As soon as the pipeline runs, the app hides the import overlay and shows a full-screen processing screen. In the middle a large progress circle runs (220 × 220 pixels) with stage icon, status text and percent number; in the background a subtle splat animation symbolically visualizes the running computation. Top left an info panel can be shown that displays live metrics from training and SfM. At the bottom there are Pause/Resume, Cancel, and in case of error a Retry button.

C-18 SplatTrainingView (background animation)

WHERE

Full-screen background behind the progress circle, hidden on cancel or error.

TECHNICAL

Decorative animation that, depending on pipeline progress (0...1), renders an increasing number of small animated splat particles. The source is a computed progress value that maps SfM phases to 0–0.2 and training to 0.2–1.0 (Frame Quality to 0–0.05). This way the splats visibly “build up” while training runs. Purely decorative — the display does not show actual intermediate results of the current training (that would be live preview in Expert Mode). On cancel or failure it is hidden and only the status circle remains visible.

IN PLAIN WORDS

In the background a small animation of dancing dots runs, so that the screen does not look so empty during computation. This is not your actual 3D model — you only see that after training in step Z3. The animation does, however, have the same tonality, so you can read from the approximate density how far training has progressed. At the start only a few dots are visible, towards the end the background fills significantly more densely — a pretty visual indicator in addition to the percent display in the circle. If the animation bothers you (e.g., because you want to work in the background alongside), you can switch to Expert Mode, where it is omitted.

C-19 Large progress circle

WHERE

Center of the processing screen, 220 × 220 pixels.

TECHNICAL

Two rings rendered on top of each other: outside a muted track ring, inside a filled progress ring with accent or red stroke (red on error). Inside the circle a stage icon (brain for training, camera for SfM, film for video frame extraction, sparkles for Frame Quality), stage title and the live animated percent number in 32-point rounded font. The icon pulsates gently as long as processing is active. The display interpolates on a 30 Hz timer smoothly towards the current actual progress — with a constant creep (0.0003/frame) plus a proportional share (4 % of the gap) and a soft ceiling that sets to 80 % of the next expected milestone (for SfM from a hardcoded milestone table). This way progress feels fluid, even if the actual SfM updates only arrive every few seconds.

IN PLAIN WORDS

The large circle in the middle is your main display while the app computes. It fills smoothly, even when the actual computation updates only come every few seconds — this gives you the feeling that something is happening, instead of staring at a frozen percent for minutes. The symbol in the middle switches depending on whether frames are currently being extracted (film icon), cameras are being aligned (camera icon), or Gaussians are being trained (brain icon). The percent number refers to the current sub-step — you see the overall pipeline in the slim bar at the very top. On error the ring turns red instead of blue, and the icon no longer pulsates, so you immediately notice that something has gone wrong.

C-22 Info Button (show metrics)

WHERE

Top left on the processing screen, 32 × 32 pixels.

TECHNICAL

Simple button with material background. Toggles the info panel on or off. Icon switches between info-circle outline and info-circle filled when active. Smooth fade-in animation. Tooltip “Show detailed processing metrics”.

IN PLAIN WORDS

By default the screen is deliberately tidy — just the large progress circle, you don’t see more at first. If as a technically interested user you want to know more precisely what is happening (which iteration, how high the loss, how many Gaussians), click the i symbol at the top left. A small panel folds out at the bottom and shows all live values. A renewed click hides it again. The setting is not persistent — on every new training run the panel is initially hidden again, which is deliberately chosen so as not to scare off beginners.

C-23 Info Panel (live metrics)

WHERE

Bottom left on the processing screen, visible only when `showProcessingInfo == true`.

TECHNICAL

Two-column panel with ultra-thin material background. Left column: stage-specific info lines — for SfM status text and percent; for training iteration, combined loss, L1 loss, D-SSIM loss, Gaussian count (colored orange), speed (it/s), elapsed time, computed ETA, SH degree and learning rate. Right column: status text, time info string, inline loss chart (see C-28) and a discoverability nudge (see C-32). All values are read from training status, which is updated on every training tick.

IN PLAIN WORDS

The info panel shows all live values that in Expert Mode would permanently be in the Inspector sidebar: current iteration, loss value (smaller = better), number of Gaussians, speed, estimated remaining time, SH degree and learning rate. On the right side a tiny loss curve additionally runs along, which tells you at a glance whether training is going in the right direction. If training seems sluggish, a glance here helps — a loss that no longer falls, or an ETA that no longer decreases, indicate problems. If the loss explodes (suddenly becomes huge) or shows NaN, training has become unstable and a Cancel + Retry or switch to a different preset is sensible.

C-25 Pause/Resume Button

WHERE

Bottom navigation bar, visible only during the training stage (NOT during SfM) and as long as processing is running.

TECHNICAL

Bordered button. Calls Pause or Resume depending on status. Label switches between “Pause” (with pause icon) and “Resume” (play icon). During the SfM step the button is not shown, because Apple Photogrammetry has no pause semantics. The pause state fully preserves iteration, Gaussian status and optimizer momentum — Resume continues where it was previously stopped.

IN PLAIN WORDS

While training is running, you can stop it anytime and resume later. Useful when you want to do something else on the Mac in between that needs a lot of GPU — e.g., video editing, game testing or a render export from another app. Click Pause, do your thing, click Resume, training continues exactly where it was. Iteration counter, Gaussian count and optimizer momentum are fully preserved, the pause state costs you no quality. During the SfM phase Pause is not available — Apple Photogrammetry has no stop function, you have to work with Cancel in an emergency there.

C-26 Cancel Button

WHERE

Bottom navigation bar, visible while processing is running (SfM or Training).

TECHNICAL

Red bordered button. Opens a confirmation dialog with title “Stop and discard progress?”, buttons “Discard Progress” (destructive) and “Keep Running” (cancel). On confirmation the cancel flag is set, the training task ended, the SfM subprocess ended if necessary, and a summary line with cancel status is written to the JSONL log. In contrast to Pause, training buffers and status are discarded.

IN PLAIN WORDS

The cancel button. In contrast to Pause this is final — if you want to restart afterwards, processing runs from the beginning, all already-trained iterations are lost. Useful when you mistook the preset, training is running far too slowly, or the app is obviously producing garbage results and you don’t want to wait. Before the actual cancel the app asks again via a confirmation dialog, so you don’t accidentally lose hours of compute time. If you only want to briefly interrupt, better take Pause.

C-27 Retry Button

WHERE

Bottom navigation bar, visible when the pipeline has failed (SfM status starts with “SfM failed” or training is in error state).

TECHNICAL

Accent button. Restarts the entire pipeline. Before starting it is checked whether imported images/videos are still present. Previous error logs are kept in the JSONL directory; a new run writes a new log file with the current timestamp.

IN PLAIN WORDS

If SfM or training aborts with an error message, you can try again here. Sometimes that helps, because many steps (RANSAC, densification) have random components and a second attempt can succeed where the first failed. The entire pipeline then runs again from the start — SfM and training, in a fresh JSONL log file. If even the second attempt fails, usually the input images are the problem (too few, too little overlap, motion blur, poor light); then go back with Back and swap your material. Tip: look in parallel at the training logs (Help → Open Training Logs), there it says in more detail where exactly it got stuck.

C-28 Inline Loss Chart

WHERE

In the info panel, right column, visible only during training with non-empty history.

TECHNICAL

Compact drawing area (40 pixels high), draws the loss history as a 1-pixel line in accent color. Data is filtered to finite values (NaN protection for unstable trainings). Min/Max are computed across the entire history — the chart thus auto-zooms to the value range. The last loss value sits at the top right above the chart. The history itself is built up in the app state at every training tick (typically every 100 iterations).

IN PLAIN WORDS

A tiny loss curve that shows you at a glance whether training is "converging" (line falls to the right) or whether it is stuck or exploding (line flat or rising). With a healthy training the line falls steeply at the start and then flattens — that is the expected course, similar to a halving curve. The chart automatically zooms to the current value range, so even small improvements at the end of training remain visible. If the line suddenly shoots up or freezes, that is a good signal that something is going wrong — either the material is problematic or a different preset would be more suitable. You find the chart in the info panel, which you show top left with the i symbol.

C-32 Discoverability Nudge (Expert Mode hint)

WHERE

In the info panel, right column at the bottom, visible only during training AND in Simple Mode.

TECHNICAL

Small line with eye icon and caption text "Switch to Expert Mode (⌘2) for live splat preview", in a restrained tone and 10-point font. No interactive element, just a hint. Does not react to click — the user has to actually press Cmd+2 or click the menu Mode → Expert Mode.

IN PLAIN WORDS

A subtle hint that in Expert Mode during training the current intermediate version of your 3D model is visible live in the viewport. In Simple Mode this is deliberately hidden to keep the UI calm — but many users don't even know that this function exists, so we point to it gently here. Press Cmd+2 and training continues in the background while you can watch how your model comes together before your eyes. This is also a good tool to estimate already after a few thousand iterations whether the result will be good, or whether you'd rather cancel and start over. Cmd+1 brings you back to the Simple view anytime.

When to move to the next stage?

The app switches automatically to Z3 (Preview) as soon as training is successfully completed — you don't have to click. The bottom navigation bar then switches from Pause/Cancel to a Back button (back to Import) and an Export button (forward to Export). In error cases (red error message, stage icon is X) Retry appears instead, and you have to decide whether to start again or to go back to Import with Back to change image material.

Z3 — Preview (rotate 3D model)

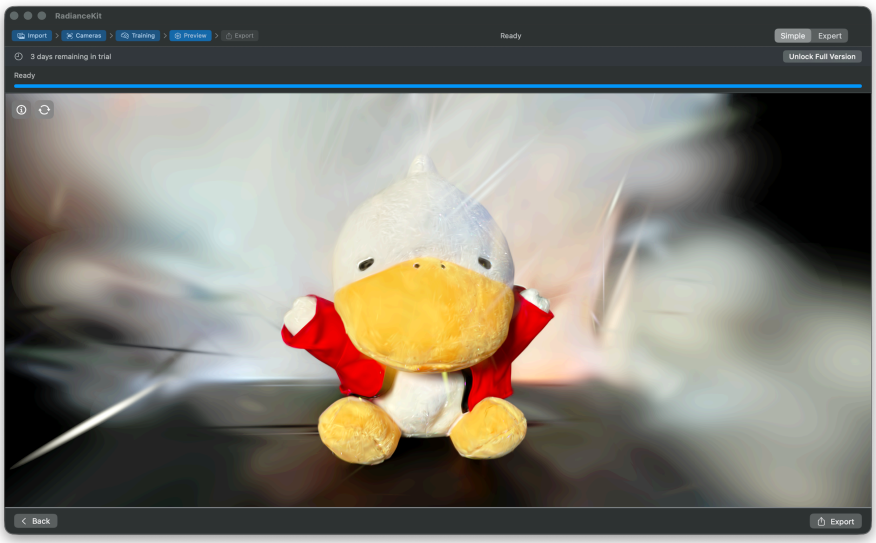

Figure 38: Simple Mode preview step with 3D viewer

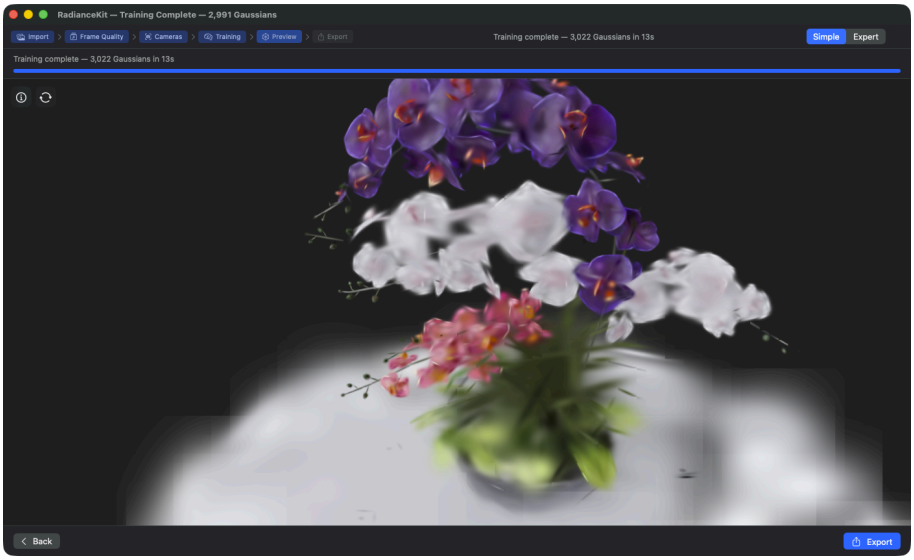

Figure 39: Z3 preview after training completion — Bjoern's Blender bouquet reconstructed, header shows "Training complete — 3,022 Gaussians in 13s", Back and Export buttons at the bottom

WHAT YOU SEE IN THE IMAGE Crumb trail marks “Preview” as active stage. Full-screen 3D viewport renders the finished trained bouquet scene (synthetic Blender test set by Bjoern, 60-frame subset from 960 hemispherical cams). Header status bar: “Training complete — 3 022 Gaussians in 13 s” — gives final Gaussian count and training time. Drag in the viewport rotates the camera (yaw/pitch); scroll wheel zooms along the view direction. “Back” button (bottom left) returns to Z2 for resume or re-run; “Export” button (bottom right, primary) navigates onward to Z4.

After completion of training, the app automatically lands in the preview. Here you see your finished Gaussian Splatting model in a fullscreen Metal view and can rotate, zoom, and pan it with mouse and trackpad. On top of the viewport sits a small overlay with camera controls and info — auto-rotation, training statistics, reset button. Before the next step (Export) it is advisable to inspect the model from different angles to make sure the reconstruction is clean.

C–36 SplatViewportView (3D main view)

WHERE

Fullscreen background of the preview step.

TECHNICAL

Metal-based 3D viewport that renders the finished point cloud. The renderer is RadianceKit’s OWN ForwardPass rasterizer — the same one that already renders the splats during training — so it is true WYSIWYG (what is trained is shown and exported exactly the same). Tile-based rendering pipeline with order-independent transparency. If the renderer cannot be initialized (e.g., because Metal is not available on the system), a black background with “Metal not available” text appears instead. The view ignores the safe area, so the model reaches to the window edge.

IN PLAIN WORDS

The main viewport. Here you see your finished 3D model reconstructed from your photos, rendered on the GPU in real time. Click and drag with the left mouse button to rotate. Scroll wheel or trackpad gesture with two fingers to zoom. Right mouse button or Cmd+drag to pan. The model consists of tens of thousands of semi-transparent 3D ellipsoids (“Gaussians”) that reconstruct your scene photorealistically — each one has a position, orientation, shape and color that training has learned. In the rare case that your Mac does not support Metal, you see instead a black background with a hint message — RadianceKit absolutely needs a Metal-capable GPU.

C-37 CameraControlsOverlay (controls overlay)

WHERE

Above the viewport, floating.

TECHNICAL

Compact UI overlay with buttons for auto-rotation (turntable), reset camera, background choice (gray/black/white), save screenshot, toggle info panel. Binds to the camera parameters (distance, azimuth, elevation, target, FOV) and controls the auto-turntable. During training (when the user in Expert Mode wants to see the viewport co-running), the overlay additionally shows a compact training status line.

IN PLAIN WORDS

The small floating bar above the model. Here you start auto-rotation (the model rotates by itself, good for screenshots and short demos), reset the camera via Reset to the start position (in case you got lost), switch the background (gray for neutral, black for maximum contrast, white for bright models), and take screenshots directly that are saved under /Pictures. Practical when you want to show a certain detail from a quite specific angle without having to extra export the entire model. Auto-rotation is also a good test of whether the model looks equally good from all sides or whether there is a "shabby side" that arose from missing shots.

C-38 Export Button (navigation bar)

WHERE

Bottom navigation bar in Z3.

TECHNICAL

Accent button with label "Export" and share icon. Click triggers the switch to Z4. Before that the parent view checks whether the full version is unlocked — if not, instead of the export stage the lock view is shown (see U-06).

IN PLAIN WORDS

When you are happy with the result, click Export and you land in the last step, where you choose the format and save. Without a purchased full version, you instead land on a screen lock with an unlock hint and a purchase button — the app does not want to push the full version on you, but Export is one of the premium features. Once you have completed the purchase, the app runs on directly in the unlocked state and you land on the familiar export stage. If you change your mind, you come back to the preview via the Back button and can continue to rotate the model.

When to move to the next stage?

Before you export, rotate the model once completely around and check: Are all areas you covered in your input images present? Are there floating “floaters” (Gaussian splat clouds floating freely in the air)? Does the background/sky look clean or smeared? Severe problems can only be fixed by re-training — either with more images, a different preset, or in Expert Mode with floater reduction settings.

Z4 — Export (choose format & save)

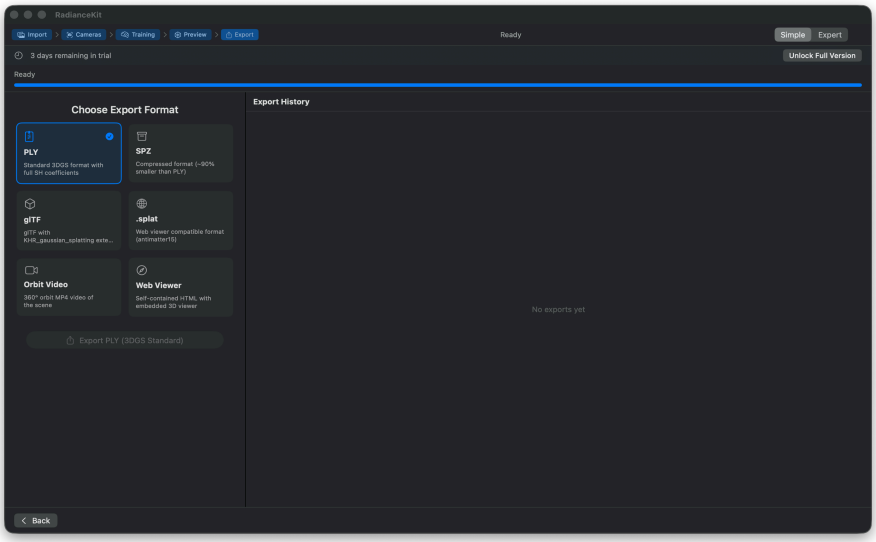

Figure 40: Simple Mode export step with format cards

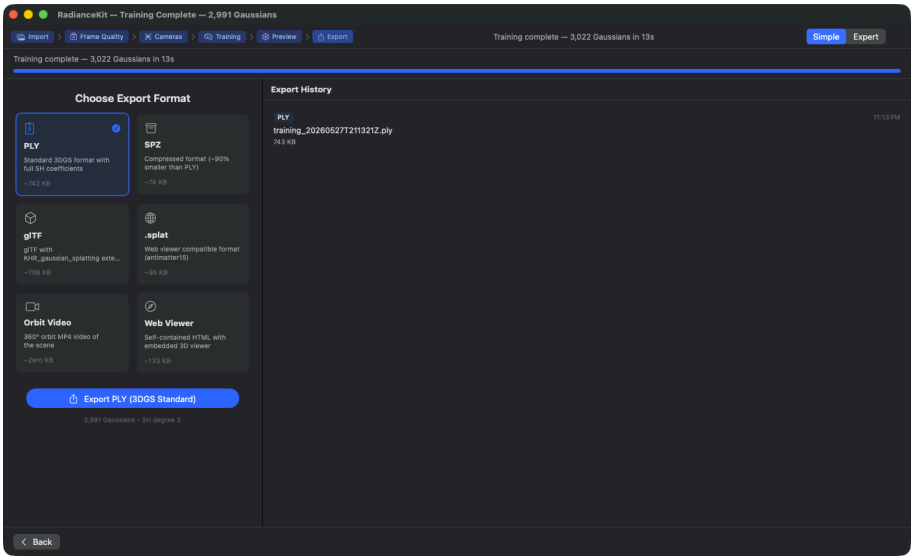

Figure 41: Z4 export cards — 6 formats (PLY 742 KB selected, SPZ 74 KB, glTF 708 KB, .splat 96 KB, Orbit Video, Web Viewer 133 KB), export history sidebar on the right with already exported PLY

WHAT YOU SEE IN THE IMAGE Crumb trail marks “Export” as active stage. Left card grid “Choose Export Format” with all six options: PLY (standard 3DGS, 742 KB, with full SH coefficients — preselected here with blue check mark), SPZ (compressed 3DGS format, ~90 % smaller than PLY, 74 KB), glTF (with `KHR_gaussian_splatting` extension, 708 KB), .splat (web-viewer compatible via antimatter15, 96 KB), Orbit Video (360° MP4 of the scene, live size computation), Web Viewer (standalone HTML with embedded 3D viewer, 133 KB). Size figures are computed live from the current Gaussian count and format overhead. On the right “Export History” lists already-completed exports with format pill, file name and timestamp — click reveals in Finder. Primary CTA bottom left: “Export PLY (3DGS Standard)” with Gaussian subtitle “2,991 Gaussians · SH degree 3”.

In the last step you choose from 6 export formats (PLY, SPZ, glTF, .splat, orbit video, web viewer) via a 2-column card grid, click Export and choose the save location in the macOS dialog. On the right a history of all previous exports runs — on card selection the estimated file size is shown immediately under each card, so you e.g. prefer SPZ if you want to go on the web (small), and PLY if you want to import into another software (SuperSplat, Postshot, Blender via plugin) (large and complete).

C–39 2-Column Format Grid

WHERE

Left main side of the export step.

TECHNICAL

Card grid with two flexible columns and 12 points spacing. Iterates over the formats offered in Simple Mode — a filtered subset of the full format list that contains only the 6 most important formats: PLY, SPZ, glTF, .splat, orbit video, web viewer. Compressed PLY and SOG are offered ONLY in Expert Mode.

IN PLAIN WORDS

A card grid with the 6 formats that are relevant in Simple Mode: PLY (standard format for other 3D tools), SPZ (compressed variant for web), glTF (official Web3D standard), .splat (for antimatter15 web viewer), orbit video (finished MP4 to show off), and Web Viewer (standalone HTML file with embedded 3D player). This way you cover 90 % of use cases. If you need one of the less common formats (compressed PLY or SOG for extreme compression), switch to Expert Mode, there all 8 formats are available. The compact selection here is intentional, so beginners are not overwhelmed by the variety.

C-40 Format Card Button

WHERE

Each card in the grid.

TECHNICAL

Simple button with card layout: icon (e.g., document zipper for PLY, archive box for SPZ, video icon for orbit video) on top, format name as headline, description caption (truncated to 2 lines), below that the estimated file size (computed live from format, Gaussian count and SH degree and formatted as KB/MB). On click the format is selected. The selected card gets an accent background, accent border and a check mark icon top right. Tooltip is the format description.

IN PLAIN WORDS

One card per format. Click one, it is highlighted with an accent color and a check mark, and the export button below adjusts its text (“Export PLY”, “Export SPZ”, etc.). Each card shows a fitting symbol, the name, a two-line short explanation, and the estimated file size for your current training result. The size helps you choose sensibly — if you want to send the result by email, take the smallest variant (usually SPZ or .splat); if you want to keep working in another 3D software, take the one with the best compatibility (typically PLY). When hovering over a card, the tooltip shows a more detailed description, in case you find the differences between the formats unclear.

C-41 Video Duration Slider

WHERE

Under the format grid, visible only when a video format is chosen (orbit video or social video).

TECHNICAL

Slider 3–30 seconds in 1-second steps, binds to the video length in app state. Maximum width 300 pixels. Only shown when a video format is selected. For non-video formats the slider is completely removed from the view — no dead space.

IN PLAIN WORDS

If you choose an orbit video as export, here you can determine the length. 3 seconds = very fast rotation, 30 seconds = slow, calm rotation around your model. For social media reels (Instagram, TikTok), usually 6–10 seconds is ideal — long enough to show the model, short enough that the viewers do not bounce off. For presentations or portfolio videos you may happily take 15–20 seconds. The slider only appears when a video format is selected; for file formats like PLY or SPZ it would be pointless and is hidden.

C-42 Export Button

WHERE

Under the format grid (and under the duration slider, if video chosen).

TECHNICAL

Large accent button. Label: “Export {format-name}”, share icon. On click the macOS save dialog opens with a format-fitting extension and default filename “scene.{ext}”; on confirmation the export is written to the chosen URL. Disabled when no training result is present or an export is already running.

IN PLAIN WORDS

Click, choose save location in the macOS dialog, done — the app writes the file in the chosen format to the selected location. Default name is “scene.{extension}” (e.g., “scene.ply” or “scene.spz”), you can change it in the dialog as you like before you save. The button is gray as long as no training result is yet present (should never happen here, because otherwise you wouldn’t be in the export step) or another export is already running. As soon as the export is running, a progress display appears under it; the app remains operable, so you can already prepare the next export.

C-43 Export Progress Bar

WHERE

Under the export button, visible only while an export is running.

TECHNICAL

Progress display with max width 300 pixels, under it caption “Exporting… N %”. The value runs from 0 to 1 and is updated during writing — with PLY in chunks of 10 000 Gaussians, with SPZ once after quantization, with orbit video in frame intervals.

IN PLAIN WORDS

While the export is running, you see here the progress as a slim bar plus percent display. PLY is usually done within seconds, because the file is simply written out binary. SPZ takes a bit longer, because the data is quantized and compressed in the process. Orbit video is the most time-consuming export — here each individual frame is re-rendered; depending on resolution and length, it can take a minute or longer. During the export the app remains operable, so you can already prepare the next format or keep clicking in the viewport.

C-44 Export Error Display

WHERE

Under the progress bar, visible only when an error occurred during the last export.

TECHNICAL

Red line with warning icon and error text. Red 8 % background opacity, rounded corners. Max width 400 pixels. Common error causes: SOG expects `cwebp` in the system PATH (not App Store compliant); write error with full disk space; sandbox error with save targets outside the allowed area.

IN PLAIN WORDS

If the export goes wrong, a brief plain-text description of the problem appears here in red. Usually the cause is obvious — no space on disk, no write permissions for the target folder, or a target location outside the sandbox-allowed areas. Specifically with the SOG format it happens that `cwebp` is missing in the system; in this case SOG is not usable and you have to fall back to SPZ. If the error message is unclear, look in the log directory (Help → Open Training Logs), there it says more thoroughly what went wrong. If in doubt, it helps to simply choose a different save location — e.g., the desktop.

C-46 Export History List

WHERE

Right side of the export step.

TECHNICAL

List over the export history (persistent as JSON in the UserDefaults, maintained after every successful export). Each row shows format badge (small, accent-colored), timestamp (HH:mm), file name (truncated to 1 line) and formatted file size. Click on a row opens Finder with the selected file. Empty state: “No exports yet”.

IN PLAIN WORDS

A list of your previous exports — format, time, file name, size, in chronological order. Click a row and the file is highlighted in Finder without you having to navigate through folders yourself. Practical when you need the last export again an hour later and no longer know where you saved it — the history remembers that. If you have never exported anything, a friendly hint “No exports yet” stands here. The list survives restarts of the app, because it is stored in the UserDefaults.

C-48 History Context Menu (right-click)

WHERE

Right-click on a history row.

TECHNICAL

Context menu on every list entry with two actions: “Reveal in Finder” (opens Finder with selected file, like the single click) and “Copy Path” (places the full file path as text in the clipboard). The latter is useful for drag-and-drop into other apps or for handover to the command line.

IN PLAIN WORDS

Right-click on a history entry opens a small menu with two actions. “Reveal in Finder” does the same as a normal click — opens Finder with the selected file, so you see it immediately. “Copy Path” places the complete file path in the clipboard, so that you can paste it e.g. in terminal commands, in other apps, or in a note. Particularly practical when you want to pass the export on to someone or open it in another program that works with path entry. Functionally a small but helpful detail that bets on Mac-typical interaction patterns.

When is the workflow complete?

After a successful export you have your 3D model as a file on disk and the history shows a new entry. There is no “Done” button — you can append any number of exports in different formats without retraining. If you want to go back to the preview (e.g., to check a camera perspective again), use the Back button in the bottom navigation bar. If you want to start a completely new scene, go via Back to Z1 and use Clear All there, or File → New Project (Cmd+⇧+N).

Switching to Expert Mode

Press Cmd+2 anytime or choose Mode → Expert Mode (`M8`). The entire state is preserved: imported images, chosen preset, running or finished training, finished point cloud, export history, even the current stage. In Expert Mode, instead of the four-step stage, the full Inspector sidebar is shown with all ~150 control fields. In particular: the Project Navigator (see Chapter 2) offers the extended image operations (minus button, backspace delete, Cmd-Z undo, Quick Look preview), the live preview in the viewport during training, as well as all loss, MCMC, densification, and Mip-Splatting parameters. Cmd+1 switches back to Simple Mode — this also loses no state.

Frequently Asked Questions

Why does my Start Processing button stay gray?

You have not yet imported any images or a video. Drag at least one file into the drop zone or use “Browse Files”. As soon as the image list on the right contains at least one entry, the button becomes active. (With only 1–2 images it does start, but SfM aborts directly with an error — see the red validation banner.)

Why is my Export button locked?

In Simple Mode there are two levels: (a) If the training pipeline is not yet finished and you have none, the button is disabled — you have to finish Z2 first. (b) If you have not yet purchased the full version (`PurchaseManager.hasAccess == false`), you see, instead of the export stage, a lock view with lock icon and “Unlock Full Version” button, which opens the purchase sheet. Quick and Preview presets allow training for free, but export is premium.

Why can’t I choose a preset?

You can choose it — but if you tap a premium preset (Balanced, Quality, MCMC variants) without a purchased full version, the picker automatically jumps back to Preview and the purchase sheet opens. Quick and Preview are the only freely usable presets.

Why is my drop zone empty and dashed-gray, even though I am dragging images in?

Probably a UTI type mismatch. The app accepts JPG, PNG, TIFF, HEIC, MP4, MOV plus the app-own splat formats. Other image formats (BMP, GIF, WebP, RAW formats) are NOT recognized. If you are sure your image type should be included, check the file name extension — the app goes primarily by extension, not by file content.

Why does SfM take so long, even though I only have 30 images?

Apple Photogrammetry does not scale linearly — with some image constellations (in-door spaces with complex textures, motion blur, poor light) it takes significantly longer than the image count suggests. If SfM still hangs after 10+ minutes with 30 images, abort and try again with better material, or switch to Expert Mode and try COLMAP/Native SfM (Cmd+2 → Inspector → Camera Alignment).

Where do I find my training logs?

Help → Open Training Logs (Cmd+⇧+L). That opens `~/Documents/RadianceKit/Logs/`. Every training session writes its own JSONL file with timestamp in the file name — the first line is the configuration, then follows a progress line every 100 iterations, the last line is the summary with final loss and success flag.

COLOPHON

Set in SF Pro · Code in SF Mono · Typst 0.14 · 22. June 2026

© 2026 Bjoern Kindler · Bischofshofener Str. 9, 82008 Unterhaching, Germany

Made with ❤ in Unterhaching